UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Andraž Vrhovec

# Varnost podatkov v oblaku

DIPLOMSKO DELO
NA INTERDISCIPLINARNEM UNIVERZITETNEM ŠTUDIJU

Mentor: prof. dr. Aleksandar Jurišić

Ljubljana, 2016

*Besedilo je oblikovano z urejevalnikom besedil LaTeX.*

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Oblačne storitve so postale del vsakdanjika, njihova uporaba pa je v strmem porastu. Oblačni sistemi hranijo ogromne količine podatkov, zato so zanimiva tarča za kriminalne združbe, saj je dovolj že en sam uspešen vdor v sistem. Za mnoge poslovne uporabnike je prav varnost podatkov eden ključnih faktorjev, ki odloča kateri deli poslovanja se smejo opravljati v oblaku in kateri ne. V diplomski nalogi tako raziščite, kakšna varnostna tveganja obstajajo pri uporabi oblaka in kako jih lahko omilimo z uporabo kriptografije.

# IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Andraž Vrhovec,

z vpisno številko 63080117,

sem avtor diplomskega dela z naslovom:

Varnost podatkov v oblaku

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom

  prof. dr. Aleksandar Jurišić

- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela

- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 31. avgust 2016 Podpis avtorja:

# Zahvala

Zahvaljujem se mentorju, za vso potrpežljivost in nasvete, sodelavcu Jaki Močniku za vsebinske in slovnične nasvete, ter staršem, za podporo tekom študija.

# Kazalo

# Povzetek

Cilj te diplomske naloge je predstaviti pregled varnosti podatkov, ki jih zaupamo oblačnim sistemom. V prvem delu se posvetimo osnovnim definicijam oblaka in varnosti ter identificiramo nekatera varnostna tveganja, ki se pojavijo ob uporabi oblaka. Drugi del se posveča kriptografiji, predvsem šifriranju, kot orodju za zagotavljanje varnosti podatkov v oblaku. Na poljuden način poskusimo predstaviti nekaj novih idej v kriptografiji, ki so se pojavile šele pred kratkim in so zelo zanimive za uporabo v oblaku. Za konec podamo še nekaj nasvetov, kako poskrbeti za varnost podatkov.

**Ključne besede:**

oblak, varnost, kriptografija, kriptosistemi, funkcionalni kriptosistemi, homomorfni kriptosistemi, šifriranje na osnovi atributov in predikatov

# Abstract

The goal of this diploma thesis is to give an overview of data security in cloud systems. The first part covers basics of cloud and security. Additionally it identifies some risks arising from the use of cloud computing. The second part focuses on cryptography, especially on ciphers, as a tool for mitigation of those risks. We try to convey some recent ideas (especially those interesting in the context of cloud computing) from the cryptographic world to the reader in an easy manner. We conclude with some tips on how to ensure security of data.

**Keywords:**

cloud, security, cryptography, cryptosystem, functional encryption, homomorphic encryption, attribute based encryption

# Poglavje 1

# Uvod

V zadnjih letih smo priča bliskovitemu porastu popularnosti računalništva v oblaku. Gartner je v začetku leta 2015 uvrstil računalništvo v oblaku med 10 strateško najpomembnejših tehnoloških trendov, ki bodo krojili prihodnost. Oblak nam omogoča udobno, cenovno ugodno delo kadarkoli in kjerkoli, tudi zunaj pisarne, saj je vse, kar potrebujemo internetni dostop in računalnik ali telefon.

Računalništvo v oblaku je posledica razvoja več različnih tehnologij, ki so skupaj omogočile cenovno učinkovite rešitve, ki so hkrati preproste za uporabo. Končni uporabniki uporabljajo oblak, ker je preprost za uporabo ter omogoča dostop do podatkov kjerkoli in kadarkoli. Na drugi strani vedno več podjetji v želji, da znižajo stroške poslovanja in povečajo sposobnost prilagajanja zahtevam trga, seli svoje poslovanje v oblak. Posledica tega premika je selitev podatkov iz fizičnih nosilcev, ki so bili tradicionalno v lasti iste osebe kot podatki, v oblak. To pomeni, da varnost hrambe svojih podatkov zaupamo tretjim osebam. Za veliko ljudi in manjših podjetij je skrb za varnost podatkov (pre)velik zalogaj, ki zahteva ustrezno tehnično izobrazbo, strojno opremo in prostor. Kadar hranimo podatke v oblaku, nič od tega ni potrebno, saj nam vse to zagotavlja ponudnik storitev v oblaku v zameno za mesečno naročnino.

Če vseeno ne želimo vseh svojih podatkov zaupati ponudniku storitve, pa

nam kriptografija ponuja nekaj rešitev, kako zavarovati podatke pred nepooblaščenimi očmi. Posebej zanimive so metode šifriranja podatkov, saj zagotavlja zaščito pred širokim naborom groženj, ki jih bomo spoznali kasneje. V nadaljevanju bomo najprej pogledali, kaj se skriva pod pojmoma "oblak" in "varnost", nato pa se bomo posvetili kriptografskim metodam varovanja podatkov.

## 1.1 Kaj je oblak?

Priljubljen internetni rek gre nekako takole: "there is no cloud, just someone else's computer". Oblak ni nič drugega kot skupek velikega števila računalnikov, ki niso v lasti uporabnika storitve v oblaku, morda niso niti v lasti ponudnika storitve, ampak jih ta najema pri tretjem ponudniku. Pojem oblak je le krovni izraz za nove načine ponudbe in uporabe storitev, zato ga moramo podrobneje razdelati in razložiti.

National Institute of Standards and Technology (NIST) opredeli oblak kot: "Računalništvo v oblaku je model, ki omogoča vseprisoten preprost dostop na zahtevo do skupnih računalniških virov (omrežja, strežniki, pomnilniški mediji, aplikacije in storitve), ki so lahko hitro pridobljeni ali sproščeni z minimalnim trudom in popolnoma samodejno, brez posredovanja ponudnika storitve. Model oblaka sestavlja pet ključnih lastnosti, trije nivoji storitve in štirje tipi oblakov." [22]

### Lastnosti oblaka

**Storitev na zahtevo**

Stranka lahko pridobi nove vire v oblaku po potrebi, lahko tudi popolnoma samodejno, brez človeškega posredovanja.

**Širok dostop preko omrežja**

Zmogljivosti oblaka so na voljo preko omrežja in do njih lahko dostopamo preko standardnih mehanizmov, kar omogoča dostop praktično

katerekoli naprave, ki se je zmožna povezati v omrežje (telefoni, tablice, računalniki, ...).

**Uporaba skupnih virov**

Ponudnikovi fizični viri so združeni v bazen virov, iz katerega stranke dobijo virtualizirane vire. Virtualizirani vir lahko prosto prehaja med fizičnimi gostitelji, zato težko govorimo o točni lokaciji vira.

**Bliskovita prožnost**

Uporabnik lahko preprosto poveča ali zmanjša zmogljivosti virov, ki jih uporablja, glede na potrebe. Pogosto ima občutek, da so mu na voljo neomejene zmogljivosti.

**Merjenje porabe in obračun glede na porabo virov**

Oblačni sistemi samodejno nadzirajo in optimizirajo porabo virov z metrikami (poraba prostora, pasovne širine, procesorskega časa ...). Poraba virov se nadzira, kontrolira in poroča, kar omogoča transparentnost za ponudnika in uporabnika.

## Nivoji storitve

**Software as a Service (SaaS)**

Ponudnik storitev uporabniku omogoča uporabo njegove aplikacije, ki teče v oblaku. Aplikacija je dosegljiva iz različnih naprav, preko recimo spletnega brskalnika ali aplikacijskega vmesnika (API). Uporabnik nima nadzora nad infrastrukturo, na kateri teče aplikacija (omrežje in strežniki), kot tudi ne nad samo aplikacijo (z izjemo konfiguracije aplikacije).

**Platform as a Service (PaaS)**

Ponudnik storitev uporabniku ponuja platformo, na katero uporabnik namesti lastne ali kupljene aplikacije, ki so prilagojene za delovanje v

oblaku. Tudi tu uporabnik nima nadzora nad infrastrukturo na kateri teče aplikacija, ima pa popolni nadzor nad aplikacijo samo.

**Infrastructure as a Service (IaaS)**

Ponudnik uporabniku omogoča postavitev osnovnih gradnikov računalniškega sistema (procesiranje, hramba podatkov, omrežje ...), iz katerih lahko uporabnik sestavi kar želi. Uporabnik sicer nima nadzora nad spodaj ležečo fizično infrastrukturo, vendar ima ves nadzor nad virtualiziranimi viri, ki jih pridobi.

## Tipi oblakov

**Zasebni oblak**

Vsa infrastruktura je namenjena uporabi znotraj ene organizacije, ki je sestavljena iz več delov. Upravljanje, vzdrževanje in lastništvo so lahko v domeni iste organizacije, tretje osebe ali kombinacije obojega. Lokacija sistema je lahko v prostorih organizacije ali pa izven njih.

**Skupni oblak**

Infrastruktura je namenjena uporabi s strani skupnosti organizacij, ki imajo skupne zahteve ali skupen cilj. Upravljanje, vzdrževanje in lastništvo je lahko v domeni ene ali večih organizacij znotraj skupnosti, tretje osebe ali kombinacije obojega. Lokacija sistema je lahko v prostorih ene od organizacij ali pa izven njih.

**Javni oblak**

Infrastruktura je namenjena uporabi s strani vseh. Ponudnik je lahko zasebno podjetje, izobraževalna ustanova ali vladna agencija. Infrastruktura se ponavadi nahaja na lokaciji ponudnika.

**Hibridni oblak**

Hibridni oblak je kombinacija dveh ali več tipov oblakov iz prvih treh točk (zasebni, javni, skupni). Oblaka sta sicer fizično ločena, vendar

skupni standardi in tehnologije omogočajo preprosto prehajanje aplikacij in podatkov iz enega v drugega.

## 1.2 Kaj je varnost?

Varnost je širok pojem, ki ga v kontekstu računalništva lahko razdelimo na tri dele. Računalniško varnost, informacijsko varnost in varovanje podatkov. V nadaljevanju jih opredelimo bolj natančno:

**Računalniška varnost (angl. computer security)** se ukvarja z varovanjem računalniških sistemov, od strojne opreme preko programske opreme do podatkov, ki se procesirajo na tej strojni opremi. Naloga računalniške varnosti je zagotoviti integriteto, dostopnost in zaupnost vseh sestavnih delov računalniškega sistema.

**Informacijska varnost (angl. information security)** se ukvarja z varovanjem podatkov, ki se obdelujejo in shranjujejo v informacijskih sistemih. Njena naloga je zaščititi podatke pred nepooblaščenim dostopom, uporabo, spreminjanjem in uničenjem z ciljem zagotavljanja integritete, dostopnosti in zaupnosti podatkov.

**Varovanje podatkov (angl. information assurance)** se ukvarja z upravljanjem tveganj, povezanih z uporabo, prenosom in procesiranjem podatkov s ciljem zagotavljanja celovitosti, dostopnosti, zaupnosti podatkov in preprečevanje tajenja.

Iz teh opisov razberemo ključne lastnosti varnega računalniškega sistema. V literaturi te ključne lastnosti imenujemo CIA triad, po začetnicah angleških besed za naslednje tri lastnosti:

**Zaupnost (angl. confidentiality)** zajema zaščito podatkov pred nepooblaščenim dostopom. Pogosti mehanizmi, ki se uporabljajo za zagotavljanje

zaupnosti podatkov so: zaščita dostopa z uporabniškim imenom in geslom, dvostopenjska overitev (two-factor auth), biometrična overitev in šifriranje podatkov.

**Celovitost (angl. integrity)** pomeni zagotavljanje pravilnosti podatkov. Ukrepi, ki zagotavljajo celovitost podatkov, preprečujejo nepooblaščenim osebam spreminjanje podatkov med prenosom ali v mirovanju. Omogočajo sledenje spremembam podatkov skozi zgodovino (revizijska sled), obnovitev podatkov iz varnostne kopije, v primeru, da pride do nepooblaščene spremembe. Mehanizmi, ki zagotavljajo celovitost podatkov so: nadzor dostopa in zgoščevalne funkcije ter digitalni podpisi.

**Razpoložljivost (angl. availability)** pomeni, da so podatki vedno na voljo pooblaščenim osebam. Glavni vzrok nerazpoložljivosti so programske in strojne napake ter napadi vrste ohromitev storitve (denial of service). Razpoložljivost podatkov se zagotavlja z rednim vzdrževanjem strojne in programske opreme in s posebnimi ukrepi za preprečevanje napadov vrste ohromitev storitve [35].

Poleg treh ključnih lastnosti, ki jih najdemo zgoraj, je Meier [21] identificiral še tri lastnosti, to so:

**Overjanje** identificira vir podatka oziroma izvor operacije, ki je lahko človek, podatek ali računalnik, in zagotovi, da je res tisti, za katerega se izdaja.

**Avtorizacija** nadzira dostop do virov glede na pravila sistema in dodeljene pravice. Avtorizacijski mehanizmi uporabnikom preprečujejo, da bi presegli svoja pooblastila.

**Preprečevanje tajenja** onemogoča uporabniku, da izpodbija oziroma zanika neko dejanje ali veljavnost svoje obveze pri pogodbi.

Vse zgoraj naštete lastnosti so splošne lastnosti varnosti in zato seveda dotaknejo tudi računalništva v oblaku. Pri uporabi storitev v oblaku, bi morali

najbolj izpostaviti vprašanje varnosti podatkov, saj le te zaupamo v hrambo ponudniku storitve.

# Poglavje 2

# Varnostna tveganja v oblaku

Zaradi vse večje priljubljenosti storitev v oblaku in posledično ogromne količine podatkov, s katerimi razpolagajo ponudniki storitev v oblaku, so ti vedno pogosteje tarča napadov kriminalnih združb. Za sisteme, ki tečejo "v oblaku", je značilno, da navzven delujejo kot en strežnik, v resnici pa tečejo na velikem številu strežnikov, ki so med seboj povezani. To pomeni da morebitni vdor v samo en strežnik napadalcu omogoča dostop na ostale strežnike, ki delujejo znotraj istega sistema. Če si sistem predstavljamo kot trdnjavo, to pomeni, da je večina zaščitnih mehanizmov skoncentrirana na obrobju, na obzidju trdnjave. Če uspemo to obzidje predreti na samo enem delčku, se lahko več ali manj prosto sprehajamo znotraj trdnjave. Pred pojavom storitev v oblaku je bilo znotraj enega obzidja le nekaj strežnikov, zato je bilo obzidje majhno in ga je bilo lažje vzdrževati. Sistem, na katerem teče storitev v oblaku pa je lahko sestavljen iz več tisoč strežnikov, kar pomeni, da se velikost obzidja močno poveča, veliko obzidje pa je tudi težko vzdrževati, zato se v njem hitro pojavijo razpoke.

Druga stvar, ki dela sisteme v oblaku zanimive za napadalce, je koncentracija podatkov. V primeru vdora v klasični strežnik, napadalec dobi dostop do veliko manjše količine podatkov kot ob vdoru v oblačni sistem. Razlog je preprost, saj na en sam strežnik fizično ni mogoče shraniti toliko podatkov,

kot na sistem, ki je sestavljen iz tisočih strežnikov.

Ranljivost računalniškega sistema je lastnost, zaradi katere je ogroženo njegovo delovanje in varnost podatkov, ki jih vsebuje. Storitve, ki delujejo v oblaku, podedujejo vse ranljivosti, ki jih imajo klasični računalniški sistemi, zaradi posebnih lastnosti oblaka, pa obstaja še nekaj groženj, specifičnih za storitve v oblaku:

**Nedovoljen dostop do upraviteljskega in programskega vmesnika**
Narava storitev v oblaku zahteva da so upraviteljski vmesniki dostopni preko interneta široki množici uporabnikov, kar posledično pomeni, da so ti vmesniki izpostavljeni napadom preko interneta.

**Ranljivosti Internetnega protokola**
Ena ključnih lastnosti oblaka je tudi široka dostopnost preko standardiziranih protokolov, zato je potrebno, ko govorimo o varnosti v oblaku, upoštevati tudi vse ranljivosti uporabljenih protokolov, kot je recimo napad vrinjenega napadalca.

**Ranljivosti obnovitve podatkov**
Zaradi uporabe skupnih virov, ki je ena ključnih lastnosti oblaka, se lahko en fizični nosilec podatkov, ki je prej pripadal enemu uporabniku, kasneje dodeli drugemu uporabniku. Če pred tem ne poskrbimo za varen izbris podatkov, lahko drug uporabnik obnovi izbrisane podatke z nosilca in tako pride do razkritja podatkov.

**Izogibanje merjenju in plačilu**
Storitve v oblaku se običajno plačujejo po porabi. Prav tako za začetek uporabe storitve praviloma vnesemo samo nekaj osnovnih podatkov in številko kreditne kartice. Napadalec nam lahko predloži ukradene podatke ali spreminja podatke o porabi, da se sam izogne plačilu.

## 2.1 Razkritje podatkov

O razkritju podatkov govorimo v primeru, da se občutljivi in zaupni podatki znajdejo na očeh nepooblaščenih oseb. To se lahko zgodi kot posledica usmerjenega napada ali zaradi človeške napake, napake v aplikaciji ali slabih varnostnih mehanizmov aplikacije. Primeri zaupnih podatkov, ki so pogosto tarča napadov so: zdravstveni podatki, finančni podatki, osebno določljivi podatki, poslovne skrivnosti in intelektualna lastnina.

Razkritje podatkov je stara in vseprisotna grožnja v vseh računalniških sistemih, vendar je zaradi prej omenjene koncentracije podatkov ta grožnja v oblačnih sistemih še večja.

Obseg škode, ki ga povzroči razkritje podatkov, je odvisen od občutljivosti podatkov, ki so bili odtujeni. V veliko državah po svetu obstajajo zakoni in predpisi, ki podjetja, ki hranijo določene tipe občutljivih podatkov, obvezuje k določenim standardom zaščite. V primeru, da pride do razkritja podatkov, lahko odgovorno podjetje doleti denarna kazen ali celo kazenska ovadba. Poleg škode, ki je bila povzročena z razkritjem podatkov in morebitno denarno kaznijo, so tu še stroški preiskave incidenta, obveščanja uporabnikov in izgube ugleda podjetja.

## 2.2 Pomanjkljivo upravljanje identitete za dostop

Površno upravljanje lahko pripelje do nenamernega razkritja uporabniških imen, gesel in ključev, ki se uporabljajo za dostop do storitev. Ponudniki storitev v oblaku upravljajo z identitetami milijonov uporabnikov in zaposlenih, zato je potrebna velika skrb, da se zagotovi pravilno upravljanje identitete skozi celoten cikel. Potrebno je poskrbeti za takojšno prekinitev veljavnosti identitete v primeru, da pride do odpovedi delovnega razmerja zaposlenega ali prekinitve pogodbe s stranko. Uporaba šibkih gesel in neuporaba dvosto-

penjske avtentikacije in neuporaba rotacije šifrirnih ključev, gesel in digitalnih potrdil so primeri slabe prakse, ki v primeru napada napadalcu olajšajo delo in povečajo razsežnosti škode, ki jo napadalec povzroči.

## 2.3 Nezaščiteni uporabniški in programski vmesniki

Ponudniki storitev v oblaku ponujajo nabor uporabniških in programskih vmesnikov, preko katerih uporabniki dostopajo do storitev ponudnika. So stična točka med uporabnikom in storitvijo in od varnosti in dostopnosti teh vmesnikov je odvisna varnost celotnega sistema. So prva linija obrambe sistema, zato je pri zasnovi teh vmesnikov potrebno misliti na varnost s pravilno implementacijo varnostnih mehanizmov kot so avtentikacija, nadzor dostopa in šifriranje. Vmesniki morajo biti zasnovani tako, da ščitijo pred namerno in nenamerno zlorabo sistema.

## 2.4 Sistemske ranljivosti

Sistemske ranljivosti so hrošči v uporabljeni programski opremi, ki jih je možno izkoristiti za napad na računalniški sistem. Ranljivosti, ki se nahajajo v jedru operacijskega sistema ali sistemskih knjižnicah, predstavljajo grožnjo za celoten sistem.

Tako kot večina ostalih predstavljenih groženj, tudi sistemske ranljivosti niso nič novega, vendar razsežnost oblačnih sistemov doda takim ranljivostim popolnoma nov pomen. Zaradi napredka tehnologije je možno na enem strežniku gostiti več virtualnih strežnikov različnih organizacij, kar pomeni da si ti virtualni strežniki delijo pomnilnik in ostale vire, kar omogoča nove načine izrabe sistemskih ranljivosti, kot je pobeg iz hipernadzornika [28].

## 2.5 Ugrabitev uporabniškega računa

Ugrabitve uporabniških računov so posledica napadov ribarjenja[1] ali izkoriščanja sistemskih pomanjkljivosti. V primeru, ko ista uporabniška imena in gesla uporabljamo za več različnih storitev, to še dodatno poveča posledice napada. Če pride do ugrabitve uporabniškega računa za storitev v oblaku, lahko napadalec izkoristi to storitev kot odskočno desko za nove napade.

## 2.6 Notranje grožnje

Tveganje, ki ga predstavljajo notranje grožnje, kot so škodoželjni zaposleni, je dokaj novo in še vedno predmet debate v industriji računalniške varnosti. Dejstvo je, da tako tveganje obstaja, predvsem pri uporabi javnega oblaka, saj v tem primeru z infrastrukturo in podatki razpolaga tretja oseba. V tem primeru uporabnik storitve nima nikakršnega mehanizma, s katerim bi lahko nadziral zaposlene pri ponudniku storitve.

## 2.7 Organizirana trajna grožnja

Organiziranje trajne grožnje so ena najnaprednejših oblik grožnje računalniški varnosti. Za napadi te oblike so ponavadi skrivajo velike organizacije z državno podporo (tajne službe, vojska). Značilnost te oblike napada je, da napadalec temeljito preuči ciljni sistem, njegovo organizacijo in šibke točke. Ta oblika napada se pogosto začne z usmerjenim izvabljanjem, to je posebna tehnika ribarjenja, ki je usmerjena proti točno določenim osebam, v našem primeru zaposleni v ciljni organizaciji. Z usmerjenim izvabljanjem napadalec poskuša pridobiti vstopno točko v ciljni sistem. Ko enkrat pridobi dostop do notranjega omrežja sistema, to uporabi kot odskočno desko za nadaljnje premike proti

[1] Način napada, kjer napadalec poskuša po elektronski poti prepričati žrtev, da mu posreduje občutljive podatke, tako da se izdaja za nekoga, ki mu uporabnik zaupa.

želenim tarčam. Cilj napada so podatki, ki se nahajajo v sistemu žrtve, pogosto so to poslovne ali državne skrivnosti. Napadalec želi čimdlje ostati skrit, saj tako pridobi dostop do večjega števila računalnikov znotraj omrežja in s tem večje količine podatkov. V želji da ostane skrit, uporablja napredne tehnike brisanja sledov, zato je take tipe napada zelo težko zaznati, saj navzven sistem deluje normalno.

Posebna oblika organiziranih trajnih groženj so kibernetska orožja. Verjetno najbolj znan med njimi je Stuxnet, ki je sabotiral Iranski jedrski program. Kibernetska orožja se od ostalih oblik napadov razlikujejo v tem, da žrtvi povzročijo dejansko fizično škodo v obliki uničene opreme ali pa celo človeške žrtve.

## 2.8 Izguba podatkov

Izguba podatkov predstavlja za večino uporabnikov in podjetji eno najhujših nočnih mor. Trajna izguba podatkov, shranjenih v oblaku ni nujno posledica napada tretje osebe, ampak je lahko tudi posledica malomarnosti, kot je recimo nenameren izbris podatkov s strani ponudnika storitve ali posledica naravne katastrofe kot je požar ali potres. V izogib trajni izgubi podatkov ponudniki storitev v oblaku uporabljajo različne mehanizme, kot so geografska redundanca in varnostno kopiranje na drugo lokacijo. Vendar v primeru, ko uporabnik podatke šifrira še preden jih pošlje v oblak in nato izgubi šifrirni ključ, noben od naštetih mehanizmov ne more preprečiti takojšnje in trajne izgube podatkov.

## 2.9 Malomarnost

Preden začnemo uporabljati storitve v oblaku, moramo preučiti vse vidike, ki jih bo taka sprememba prinesla. Vodilni kadri v podjetjih pogosto nimajo dovolj poglobljenega poznavanja oblaka s tehnološkega vidika, kot tudi ne prav-

nih zank, ki jih vsebujejo "pogoji uporabe" določenih ponudnikov storitev v oblaku. V primeru, da pred uporabo storitev v oblaku ne naredimo temeljite analize, lahko pride do tehnoloških, pravnih, skladenjskih in poslovnih tveganj, ki ogrozijo uspeh podjetja.

## 2.10 Zloraba in zlonamerna uporaba sistemov

Pogosto ponudniki storitev v oblaku omogočajo brezplačne preizkusne račune za omejeno časovno obdobje, če pa jih ne, lahko napadalec z ukradenimi bančnimi podatki ustvari večjo količino uporabniških računov. Te potem izkoristi za izvajanje različnih nelegalnih aktivnosti. Lažni uporabniški računi napadalcu omogočajo pošiljanje neželene pošte, izvajanje napadov ribarjenja ali napadov vrste ohromitev storitve, ki so lahko usmerjeni k ponudniku storitve, njegovim strankam ali pa tretjim osebam.

## 2.11 Napadi vrste ohromitev storitve

Napad vrste ohromitev storitve je ena najpreprostejših in pogosto zelo učinkovitih oblik napada na računalniške sisteme. Cilj napada je uporabnikom preprečiti normalen dostop do storitve, kar napadalec doseže s tem, da napaden sistem prisili v nerazumno veliko porabo različnih virov, kot so pasovna širina, procesorski čas ali podatkovni prostor. Najbolj osnovna oblika napada vrste ohromitev storitve se zanaša na dejstvo da napadalec razpolaga z večjo količino določenega vira (pasovna širina, procesorski čas ...) kot napaden sistem. Obstajajo pa tudi bolj prefinjene oblike napada, kjer napadalec z veliko manjšo količino razpoložljivih virov izrabi varnostno pomanjkljivosti v sistemu ali aplikaciji in onemogoči tarčo napada. Napadi vrste ohromitev storitve običajno ne pustijo trajnih posledic, vendar predstavljajo veliko neprijetnost za uporabnike, saj le ti za čas trajanja napada ne morejo uporabljati storitve, prav tako pa škodujejo ugledu ponudnika.

## 2.12 Ranljivosti zaradi uporabe skupnih virov

Zadnji sklop varnostnih groženj v oblaku predstavljajo ranljivosti, ki so posledica ene od ključnih lastnosti oblaka, to je uporaba skupnih virov. Ponudniki storitev v oblaku zagotavljajo navidezno "neomejene" vire končnemu uporabniku z uporabo virtualizacije in skupne rabe virov. Pri tem uporabljajo klasično strojno opremo, ki sicer s pomočjo posebne programske opreme omogoča delitev virov, vendar ni bila zasnovana, da bi omogočala varno skupno uporabo. Tak tip napadov izkorišča dejstvo, da si različni virtualni strežniki delijo fizično strojno opremo. Zato je možno s posebno obliko napada, ki se imenuje napad preko stranskega kanala, pridobiti informacije iz žrtvinega strežnika, brez da bi dejansko prišlo do vdora. V času pisanja te diplome še ni prišlo do takega tipa vdora v praksi, vendar obstajajo primeri takih napadov, ki so jih demonstrirali pod laboratorijskimi pogoji.

# Poglavje 3

# Osnove kriptografije

Kriptografija omogoča doseganje informacijske varnosti s pomočjo matematičnih postopkov. Verjetno najbolj znan tak postopek je šifriranje, ki zagotavlja zaupnost podatkov, pogled tega v področje kriptografije spadajo tudi digitalni podpisi, avtentikacijske kode MAC in zgoščevalne funkcije, ki nam pomagajo pri ohranjanju celovitosti in overjanju identitete podatkov. Poleg tega se kriptografija ukvarja tudi z razvojem varnih generatorjev naključnih števil, ki so pomembna osnova za generiranje ključev in varno implementacijo kriptosistemov.

Šifriranje je način, kako skriti vsebino sporočila (čistopis) pred nepooblaščenimi očmi. To dosežemo z uporabo algoritma, ki mu kot vhodni podatek damo čistopis in šifrirni ključ, le-ta pa nam vrne tajnopis. Tajnopis vsebuje enake informacije kot čistopis, vendar ima to lastnost, da ga je mogoče prebrati le v primeru, da posedujete ključ za dešifriranje. Brez ključa je iz tajnopisa praktično nemogoče razbrati vsebino originalnega sporočila.[1]

To pomeni, da lahko tajnopis javno objavimo, varovati je potrebno samo šifrirni ključ. Ta lastnost nam omogoča, da varno poslujemo s svojo banko, pošiljamo e-pošto, ki jo lahko prebere samo prejemnik, navkljub temu, da internet ni varen komunikacijski kanal in da na njem mrgoli zlobnežev, ki

[1] Težanost je odvisna od moči kriptografske sheme in pravilnosti implementacije

želijo priti do naših zaupnih podatkov.

Potrebno se je zavedati, da so formalne zahteve za popolno varnost zelo visoke, zato se v praksi največkrat uporabi princip računske varnosti, kar pomeni, da varnost kriptosistema sloni na predpostavki, da so računske zahteve za razbitje kriptografske sheme veliko višje od tistih, ki jih ima na voljo potencialni napadalec. Najbolj znana implementacija popolne varnosti je sicer enkratni ščit, ki ga je leta 1918 patentiral Gilbert S. Vernam [33]. Claude Shannon je v leta 1949 objavljenem poročilu [29] dokazal, da le-ta zagotavlja popolno varnost. Kot zanimivost lahko povemo, da je domnevno do enakih zaključkov nekaj let prej prišel tudi ruski znanstvenik Vladimir Kotelnikov [25], vendar njegovo poročilo še vedno ostaja zaupno. Problem enkratnega ščita je v tem, da zahteva ključ enake dolžine kot sporočilo, ki ga želimo šifrirati. Prav tako lahko vsak ključ uporabimo samo enkrat, v nasprotnem primeru pa sistem ne zagotavlja več popolne varnosti.

**Kriptografska shema**

**Definicija 3.1.** Kriptografska shema je peterka $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, kjer veljajo naslednje lastnosti:

1. $\mathcal{P}$ je končna množica možnih čistopisov.
2. $\mathcal{C}$ je končna množica možnih tajnopisov.
3. $\mathcal{K}$ je končna množica možnih ključev.
4. Ta vsak $k \in \mathcal{K}$ obstaja šifrirni postopek $e_K \in \mathcal{E}$ in pripadajoč dešifrirni postopek $d_K \in \mathcal{D}$. Za funkciji $e_K : \mathcal{P} \to \mathcal{C}$ in $d_K : \mathcal{C} \to \mathcal{P}$ velja $d_k(e_k(x)) = x$ za vsak $x \in \mathcal{P}$.

Matematično definicijo kriptosistema, podano v definiciji 3.1, lahko povzamemo z naslednjimi besedami. Kriptografsko shemo sestavlja pet sestavin. Čistopisi, tajnopisi, ključi, šifrirni in dešifrirni postopek. Pomembno je, da ob

danem ključu in šifrirnem postopku, obstaja dešifrirni postopek, ki nam vsak tajnopis, ki je bil šifriran z izbranim šifrirnim postopkom, nedvoumno pretvori nazaj v čistopis.

Skozi zgodovino človeštva se je zelo zgodaj pojavila želja po skrivanju sporočil in s tem so se razvile tudi prve metode šifriranja, vendar nobena od njih ni prestala preizkusa časa.

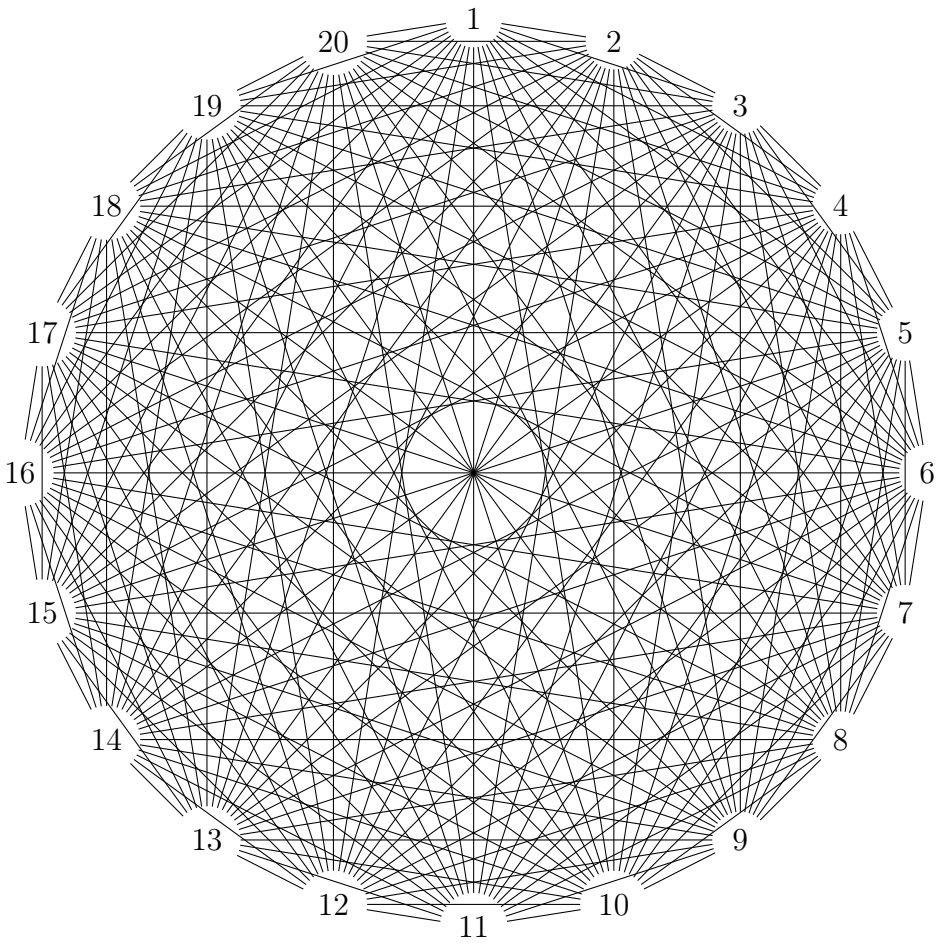
Šifrirne algoritme delimo na dve skupini:

1. simetrične oziroma klasične šifre,

2. asimetrične šifre oziroma kriptosistemi z javnimi ključi.

Simetrične šifre so najstarejša in najbolj razširjena vrsta šifriranja, kamor spadajo tudi vse zgodovinske šifre, ki se ne uporabljajo več. Poglavitna lastnost simetričnih šifer je, da se za šifriranje in dešifriranje uporablja enak ključ. Take šifre so lahko zelo preproste in hitre, vendar se pojavi problem distribucije ključev. Tajnopise lahko prenašamo po nezavarovanih kanalih, kar pa ne velja za ključe. Zato si mora vsak par uporabnikov, če želi med sabo komunicirati v šifrirani obliki, najprej po varnem kanalu izmenjati šifrirni ključ. Število potrebnih izmenjav raste $O(n^2)$ glede na število uporabnikov sistema, in na sliki 3.1 vidimo, da je že pri dvajsetih uporabnikih število potrebnih izmenjav zelo veliko.

Ta problem rešijo asimetrične šifre, ki uporabljajo en ključ za šifriranje, drug ključ pa za dešifriranje. Tak par ključev imenujemo javni in zasebni ključ. Kot že ime pove, javni ključ ni nobena skrivnost in ga lahko brez skrbi objavimo na javnem mestu, obratno pa velja za zasebni ključ. Sporočilo, šifrirano z javnim ključem je možno dešifrirati samo s pripadajočim zasebnim ključem. Slabost asimetričnih šifer je njihova hitrost, saj z njimi ni praktično šifrirati večjih količin podatkov. Zato se v praksi največkrat uporabljajo protokoli, ki uporabljajo obe družini šifer, kot bomo videli kasneje.




Slika 3.1: Problem distribucije ključev

## 3.1 Simetrične šifre

V kategorijo simetričnih šifer spadajo vse šifre, pri katerih se za šifriranje in dešifriranje uporablja enak ključ ali pa obstaja bijektivna preslikava, ki nam šifrirni ključ preslika v dešifrirni in obratno. Simetrične šifre še naprej delimo na tokovne in bločne šifre. Poglavitna razlika med tokovnimi in bločnimi šiframi je v načinu šifriranja podatkov.

Naj bo dano sporočilo $X$ dolžine $\ell$ ($|X| = \ell$), in bločna šifra z velikostjo bloka $w$. Sporočilo lahko razdelimo na $n = \lceil \ell/w \rceil$ blokov, kjer simbol $\|$

predstavlja združevanje (lepljenje) dveh nizov.

$$X = x_0 \| x_1 \| x_2 \| \dots \| x_{n-1}.$$

Bločna šifra vzame blok $x_i \in \mathcal{P}$ in ga šifrira s šifrirnim postopkom $e_k \in \mathcal{E}$, kjer $k \in \mathcal{K}$ predstavlja skrivni ključ. Rezultat te operacije je tajnopis bloka $x_i$, ki ga označimo kot $y_i \in \mathcal{C}$. Tajnopis celotnega sporočila $X$ je $Y$ in pridobimo ga po naslednjem postopku:

$$y_j = e_k(x_j), \quad j = 0, 1, \dots, n-1,$$
$$Y = y_0 \| y_1 \| \dots \| y_n.$$

Bločne šifre delujejo na bloku podatkov fiksne velikosti (običajno velikost ključa določi velikost bloka) in v kolikor dolžina sporočila ni deljiva z $m$, je potrebno sporočilo dopolniti s posebnim zaporedjem znakov, dokler dolžina ni večkratnik števila $m$.

Drugačen pristop pa uporabljajo tokovne šifre, ki iz ključa $k \in \mathcal{K}$ ustvarijo naključno zaporedje bitov $Z$, enake dolžine kot je sporočilo. Ključ nato zmešajo s čistopisom, tipično z operacijo XOR (ekskluzivni ali, predstavljen s simbolom $\oplus$). XOR deluje na nivoju bitov, zato take šifre lahko šifrirajo čistopis bit po bit in tako ni potrebe po dopolnjevanju.

$$Z = z_0 \| z_1 \| z_2 \| \dots \| z_{l-1},$$
$$Y = e_Z(X) = X \oplus Z.$$

### DES in 3DES

Data encryption standard (DES) je bil eden prvih modernih šifrirnih algoritmov. Originalna implementacija je nosila ime Lucifer in bila razvita v sedemdesetih letih v laboratorijih podjetja IBM. Kasneje je bila rahlo modificirana

različica tega algoritma sprejeta s strani NIST (takrat se je imenoval še NBS) kot standard DES.

DES je simetrična bločna šifra, ki deluje na blokih velikosti 64 bitov in uporablja 64-bitni ključ, vendar je v ključu možno poljubno izbrati le 56 bitov, preostalih 8 pa se uporablja samo za odkrivanje napak v ključu.

Algoritem je sestavljen iz 16 identičnih korakov oziroma krogov, nekaj pred- in po-procesiranja vhodnih podatkov, Feistelove F-funkcije ter funkcije, ki iz glavnega ključa ustvari zaporedje podključev. Na začetku se 64-bitni blok razdeli na dve 32-bitni polovici, levo in desno, ki se po vsakem končanem krogu algoritma zamenjata. Taka shema delovanja je znana tudi kot Feistlova šifra. V vsakem krogu se leva polovica bloka z operacijo XOR zmeša z rezultatom Feistelove F-funkcije, ki kot vhodne parametre dobi podključ trenutnega kroga in desno polovico bloka. Po koncu vsakega kroga se leva in desna polovica zamenjata (razen v zadnjem krogu).

Ob prelomu tisočletja se je izkazalo, da je 64-bitni ključ prešibak, da bi zadržal napade s surovo silo. Zato se je pojavila varianta algoritma imenovana 3DES. Slednji uporablja 3 ključe, pri čemer imamo na voljo 3 opcije za izbiro ključev.

1. Vse tri ključe izberemo neodvisno.
2. Izberemo ključa $K_1$ in $K_2$ in $K_3 = K_1$.
3. Vsi ključi so isti, torej $K_1 = K_2 = K_3$.

Postopek šifriranja in dešifriranja je nato:

$$C = E_{K_3}(D_{K_2}(E_{K_1}(P))),$$
$$P = D_{K_1}(E_{K_2}(D_{K_3}(C))),$$

kjer je $C$ tajnopis in $P$ čistopis.

### AES

Advanced encryption standard (AES) je bil leta 2001 izbran kot naslednik standarda DES. Med 15 prijavljenimi algoritmi je NIST izbral družino algoritmov Rijndael. Kot standard AES so bile sprejete tri različice algoritma. Vse tri uporabljajo 128-bitne bloke, razlikujejo pa se v velikosti ključa. Na voljo so 3 opcije: 128, 192 ali 256-bitni ključ.

AES se po strukturi algoritma razlikuje od DES, saj namesto Feistelove šifre uporablja substitucijsko-permutacijsko omrežje.

## Načini delovanja bločnih šifer

Bločna šifra sama po sebi kot vhodni podatek lahko sprejme samo blok točno določene dolžine, kar ni najbolj praktično, saj so sporočila pogosto različnih in predvsem nepredvidljivih velikosti. Poleg tega, če isti blok čistopisa večkrat šifriramo z enakim ključem, dobimo vsakič isti tajnopis, kar je nezaželjen rezultat, saj morebitni napadalec lahko zazna vzorce v tajnopisu in iz tega poskuša razbrati pomen. Ta problem lahko rešimo tako, da uporabimo bločno šifro v enem od načinov delovanja.

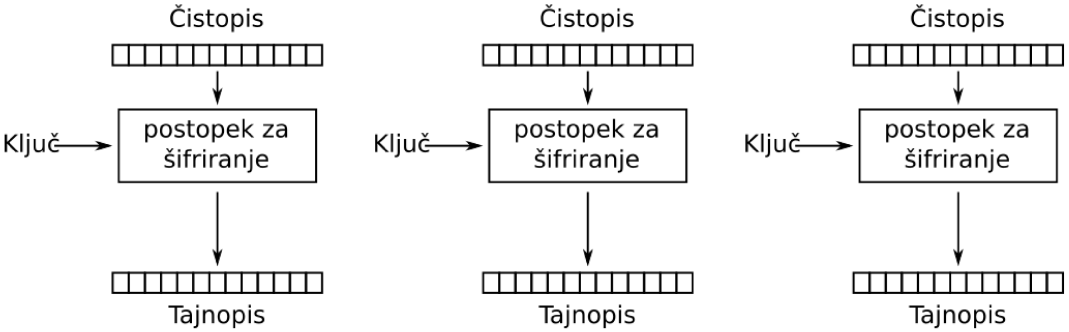
Način delovanja je algoritem, ki pove, kako vhodno sporočilo razdeliti na bloke primerne velikosti in na kak način na teh blokih potem uporabiti bločno šifro.

Večina načinov delovanja poleg bločne šifre in šifrirnega ključa potrebuje še Inicializacijski vektor (IV). Slednji skrbi, da tudi v primeru, da večkrat šifriramo isti čistopis z enakim ključem, dobimo kot rezultat različne tajnopise.

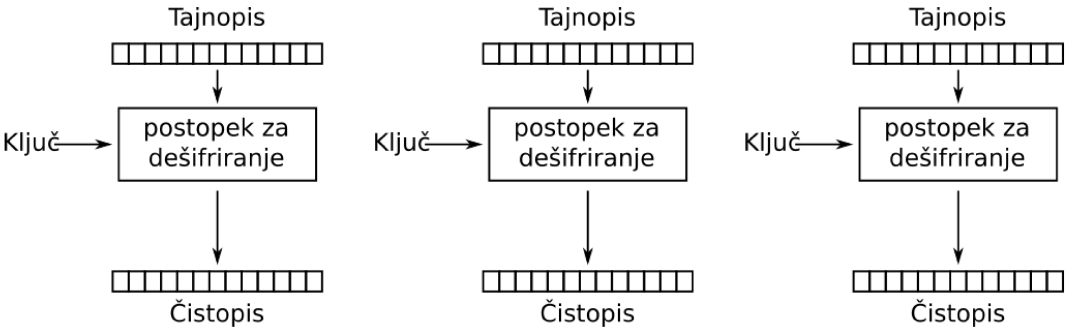
Kot smo že povedali, bločne šifre sprejemajo le bloke točno določene velikosti, zato je potrebno tudi v nekaterih načinih delovanja šifre sporočilo najprej dopolniti, tako da je njegova dolžina večkratnik velikosti bloka. Vendar obstajajo tudi načini delovanja, kjer to ni potrebno, saj ti iz bločne šifre naredijo tokovno šifro.

### ECB

Najpreprostejši način delovanja je Electronic Codebook (ECB). V tem načinu delovanja se sporočilo razdeli na bloke take velikosti, kot jih zahteva bločna šifra. Po potrebi se zadnji blok dopolni do zahtevane dolžine. Nato se vsak blok šifrira neodvisno od drugih, kar sicer pomeni, da je možno operaciji šifriranja in dešifriranja na precej preprost način paralelizirati in s tem pohitriti. Prav tako je v tem načinu delovanja možno dešifrirati poljubni blok sporočila, ne da najprej dešifriramo vse predhodne bloke.



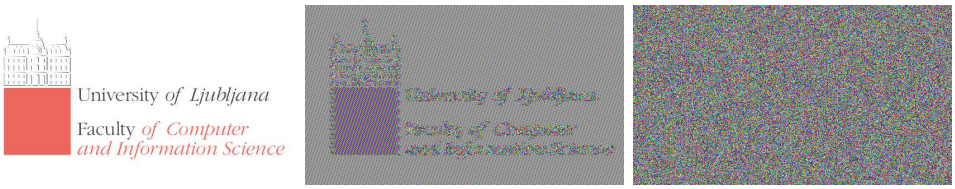

Slika 3.2: Šifriranje v načinu ECB




Slika 3.3: Dešifriranje v načinu ECB

Vse so sicer dobrodošle lastnosti, vendar ima ECB nekaj resnih pomanjkljivosti, zaradi katerih ni primeren za uporabo v resničnem svetu, ampak služi bolj kot opomin, da močni šifrirni algoritmi kot je AES, ne zagotavljajo varnosti, če so uporabljeni na napačen način.

Slika 3.4 prikazuje uporabo 128 bitne AES šifre v načinu ECB. Na šifrirani sliki je še vedno možno razpoznati logotip Fakultete za računalništvo in informatiko, kar pomeni da tak način šifriranja ne zagotavlja varnosti.




(a) Original (b) AES-128-ECB (c) AES-128-CBC

Slika 3.4: Primer različnih načinov šifriranja, uporabljenih na logotipu FRI

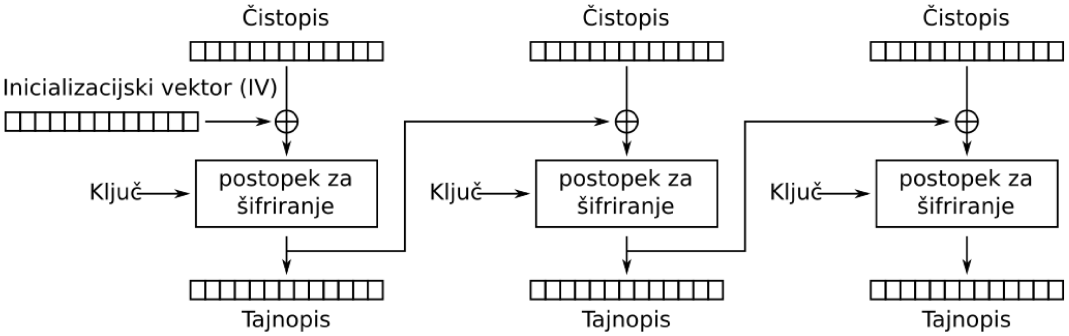
### CBC

V načinu Cipher Block Chaining (CBC) se vsak blok čistopisa pred šifriranjem z operacijo XOR zmeša z tajnopisom prejšnjega bloka. Tak pristop naredi vsak naslednji blok odvisen od vseh prejšnjih blokov. S tem se rešimo problema ponavljajočih se vzorcev v tajnopisu, kot smo jih videli v načinu ECB (slika 3.4). Posebni primer je prvi blok, saj tu nimamo na voljo tajnopisa prejšnjega bloka, s katerim bi izvedli operacijo XOR, zato namesto tajnopisa prejšnjega bloka uporabimo Inicializacijski vektor (IV). Dodatno nam naključni IV doda tudi element naključnosti v tajnopis, kar pomeni, da tudi, če šifriramo enako sporočilo z enakim ključem, tajnopis ne bo enak, saj smo uporabili drugačen IV.

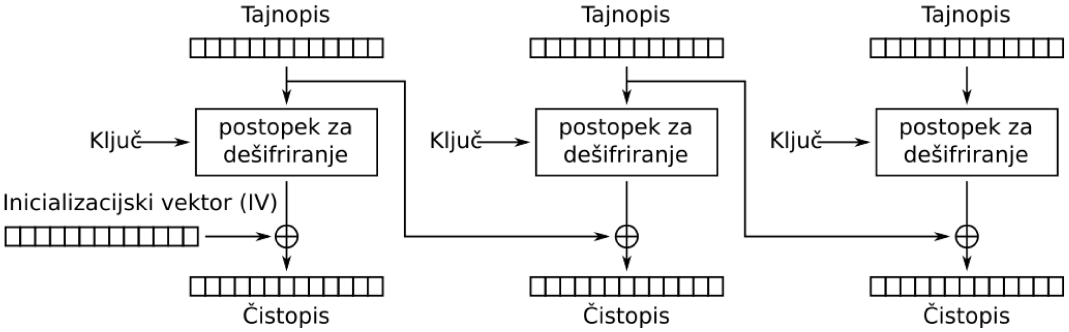
Slabost takega pristopa pa je, da je operacija šifriranja strogo zaporedna, saj naslednjega bloka ne moremo šifrirati, dokler ne končamo šifriranja trenutnega bloka.

### CFB

V Cipher Feedback (CFB) načinu delovanja se bločna šifra uporablja kot tokovna. Prednost takega načina delovanja je, da ni potrebe po dopolnjevanju sporočila, prav tako pa se uporablja samo ena smer delovanja bločne šifre (to




Slika 3.5: Šifriranje v načinu CBC



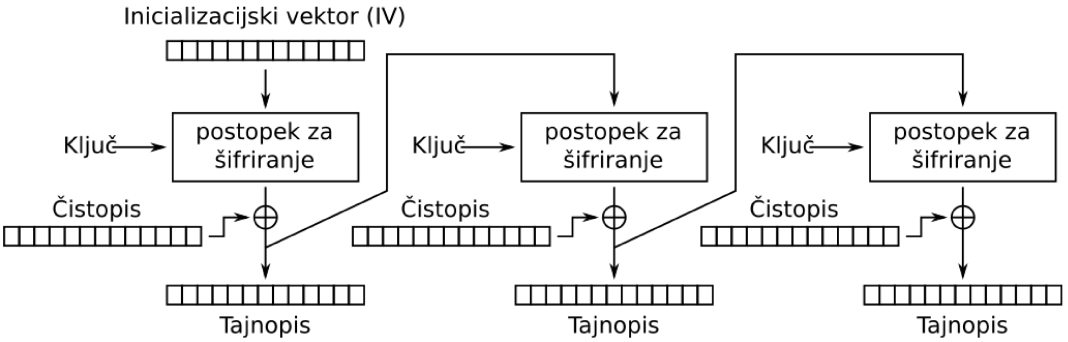

Slika 3.6: Dešifriranje v načinu CBC

je ponavadi smer šifriranja), kar poenostavi implementacijo takega načina v strojni opremi.

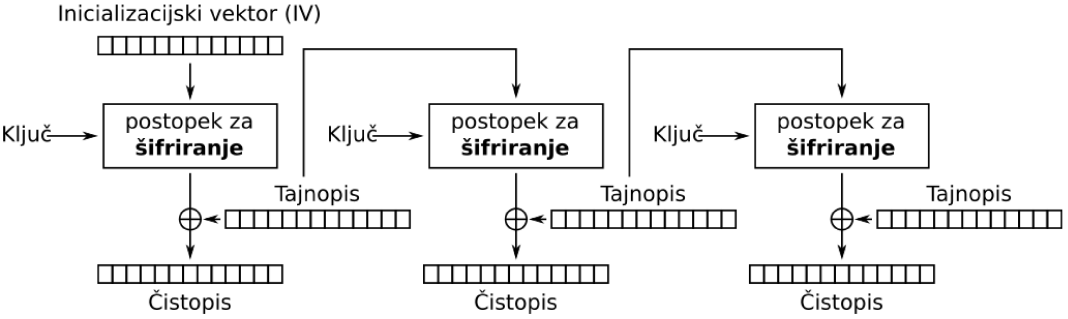
Kot se vidi na slikah 3.7 in 3.8, se v tem načinu delovanja bločna šifra uporablja le za generiranje naključnega zaporedja, tajnopis pa je rezultat operacije XOR med čistopisom in naključnim zaporedjem, podobno kot pri enkratnem ščitu (angl. one time pad).

### OFB

Output Feedback (OFB) je podoben načinu CFB, s to razliko, da se za ustvarjanje naključnega zaporedja uporablja samo IV. V primeru, da šifriramo daljša sporočila, je to lahko problematično, saj lahko pride do cikla v naključnem zaporedju. To se zgodi v primeru, da se v naključnem zaporedju pojavi blok,

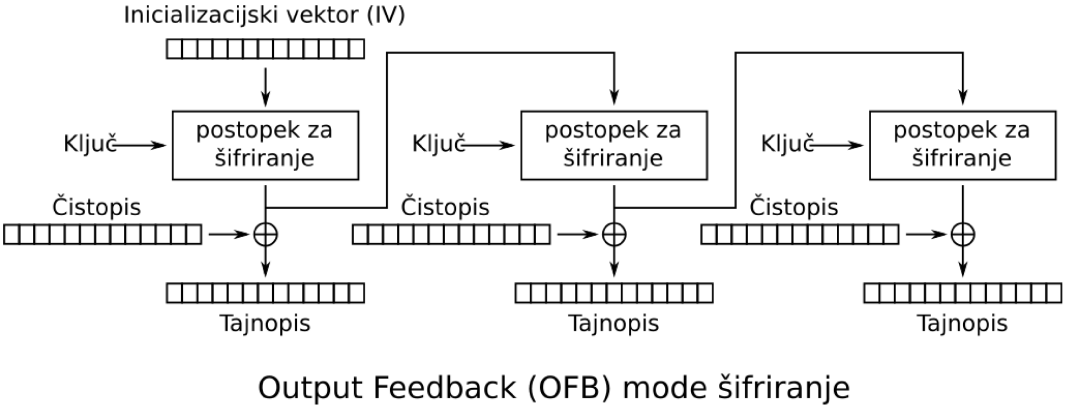



Slika 3.7: Šifriranje v načinu CFB

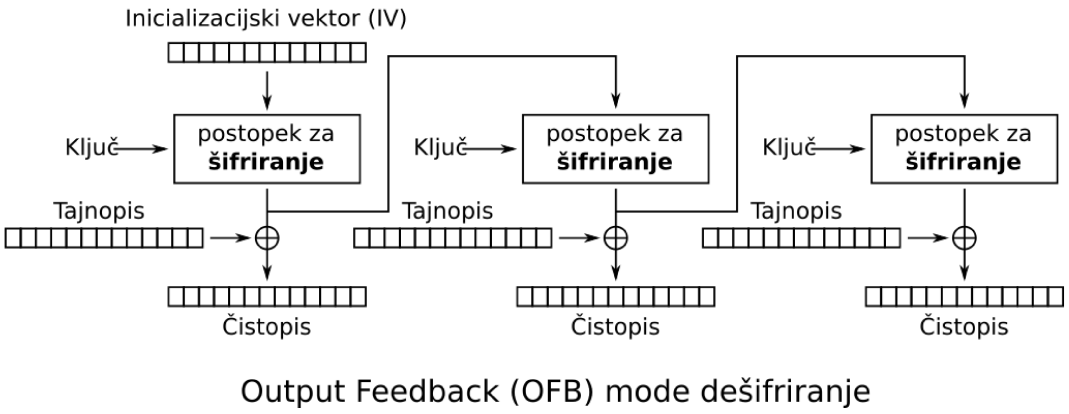



Slika 3.8: Dešifriranje v načinu CFB

katerega tajnopis je enak IV.




Slika 3.9: Šifriranje v načinu OFB




Slika 3.10: Dešifriranje v načinu OFB

## 3.2 Asimetrični kriptosistemi

Eden večjih problemov simetričnih šifrirnih algoritmov je distribucija ključev. V kolikor želita dve osebi svojo komunikacijo zaščititi z uporabo simetričnega šifrirnega algoritma, si morata najprej izmenjati ključ. Dobra praksa je tudi, da se izognemo ponovni uporabi istega ključa, kar v praksi pomeni, da si moramo z vsako osebo, s katero želimo komunicirati, izmenjati ključ in hitro se izkaže, da omrežje $n$ ljudi potrebuje $\frac{n(n-1)}{2}$ ključev. Če to preslikamo na omrežje kot je internet, ki ga uporablja približno 40 odstotkov svetovne populacije in vzamemo pesimistično oceno, da si vsak uporabnik interneta lasti samo eno elektronsko napravo, ki uporablja varno komunikacijo in tako omrežje potrebuje $\approx 4 * 10^{18}$ izmenjav ključa.

Rešitev za ta problem so lahko asimetrični kriptosistemi, oziroma kriptografija javnih ključev. Taki sistemi ne uporabljajo samo enega, ampak dva ključa. Javni ključ poznajo vsi in ga uporabijo za šifriranje sporočil. Sporočila, šifrirana z javnim ključem, lahko dešifrira samo imetnik zasebnega ključa.

Asimetrični kriptosistemi slonijo na predpostavki, da so nekateri matematični v eno smer preprosto izračunljivi, v drugo pa preprosto pretežki, da bi jih bilo možno rešiti v razumnem časovnem okvirju. Eden takih problemov je faktorizacija števil. Če si izmislite dve dokaj veliki praštevili $p$ in $q$, ju med

seboj pomnožite, dobite novo število $n$, za katero ni tako preprosto ugotoviti iz katerih faktorjev je sestavljeno, ob predpostavki, da ne poznamo $p$ in $q$.

Na žalost so asimetrični kriptosistemi, tudi zaradi svojega ozadja v netrivialnih matematičnih problemih, veliko počasnejši od simetričnih. V praksi se zato pogosto uporablja kombinacija obeh tipov sistemov. V prvi fazi se za dogovor o ključu uporabi asimetrični kriptosistem, ko pa se obe strani dogovorita o skupnem ključu, pa preklopita na simetrični kriptosistem.

## Diffe-Hellmanov dogovor o ključu

Diffe-Hellmanov dogovor o ključu je bil prvi javno objavljen algoritem, ki uporablja princip javnih in zasebnih ključev. V praksi omogoča, da se dva neznanca preko nezavarovanega kanala kot je internet dogovorita za skupni ključ, ki ga bosta uporabljala za nadaljnjo komunikacijo. Prva sta ga objavila W. Diffie in M. Hellman leta 1976 [9], vendar se je kasneje izkazalo, da je angleška tajna služba odkrila princip kriptografije javnih ključev že leto poprej (to ostalo skrito vse do leta 1997).

Metoda temelji na reševanju problema diskretnega logaritma, za katerega matematiki verjamejo, da ne obstaja algoritem, ki ga bi bil sposoben rešiti v sprejemljivih časovnih okvirjih (izjema so algoritmi, ki tečejo na kvantnih računalnikih, vendar je to povsem drug svet).

Originalna različica algoritma uporablja multiplikativno grupo števil po modulu $p$, kjer je $p$ praštevilo in $g$ primitivni koren števila $p$.

Recimo, da se želita Anita in Bojan preko interneta dogovoriti o skupnem ključu, ki ga bosta uporabljala za nadaljnjo komunikacijo. To lahko storita po postopku, ki je opisan v definiciji 3.2.

**Diffe-Hellmanov dogovor o ključu**

**Definicija 3.2.** Predstavimo nekaj preprostih korakov, kako se lahko Anita in Bojan dogovorita o skupnem ključu, brez da, bi ga razkrila komurkoli.

1. Anita in Bojan se dogovorita o skupnih parametrih, ki jih bosta uporabljala v algoritmu. To je praštevilo $p$ in primitivni koren praštevila $g$.
2. Anita si izbere skrivno število $a$, izračuna $A = g^a \mod p$ in pošlje $A$ Bojanu.
3. Bojan si izbere skrivno število $b$, izračuna $B = g^b \mod p$ in pošlje $B$ Aniti.
4. Anita izračuna $s = B^a \mod p$.
5. Bojan izračuna $s = A^b \mod p$.
6. Anita in Bojan sta dobila enak rezultat $s$, ki ga sedaj lahko uporabita kot skupni ključ za nadaljnjo komunikacijo.

Oba sta izračunala enak rezultat, ker velja:

$$A^b \equiv g^{ab} \equiv g^{ba} \equiv B^a \pmod{p}.$$

## RSA

Medtem, ko prej opisana metoda omogoča samo dogovor o skupnem ključu, pa je bil Rivest-Shamir-Adleman (RSA) prvi pravi kriptosistem, ki je uporabljal javne ključe. Varnost RSA temelji na tem, da je faktorizacija produkta dveh velikih praštevil težko rešljiv problem za današnje računalnike.

**RSA kriptosistem**

**Definicija 3.3.** Javni ključ je par $(n, e)$, zasebni ključ pa par $(n, d)$, ki ju dobimo po naslednjem postopku:

1. Izberemo dve naključni, približno enako veliki praštevili $p$ in $q$.
2. Izračunamo modul $n = pq$.
3. Izračunamo $\varphi(n) = \varphi(p)\varphi(q) = (p-1)(q-1)$.
4. Izberemo število $e$, tako da $1 < e < \varphi(n)$ tako da je $e$ tuje $\varphi(n)$.
5. Izračunamo $d$, tako da $d \equiv e^{-1} \pmod{\varphi(n)}$.

Poleg zasebnega ključa moramo moramo paziti tudi na zasebnost vrednosti $p$, $q$ in $\varphi(n)$.
Sporočilo $m \in \mathbb{Z}$ šifriramo po naslednjem postopku:

$$c = e(m) = m^e \mod n.$$

Dešifriranje je podobna operacija, le da za eksponent vzamemo zasebni ključ:

$$m = d(c) = c^d \mod n.$$

Definicija 3.3 ponazori osnovno idejo kriptosistema RSA, vendar je potrebno še nekaj korakov, da jo dopolnimo do varnega kriptosistema. Eden od problemov, ki jih lahko hitro opazimo, da lahko napadalec iz tajnopisa $c = e(m)$, za poljuben $t$, izračuna tajnopis za sporočilo $mt$.

$$e(m) * t^e \mod n = m^e t^e \mod n = (mt)^e \mod n = e(mt).$$

Tej lastnosti rečemo gnetljivost (angl. malleability), saj napadalec lahko samo s poznavanjem javnih parametrov kriptosistema in tajnopisa $c$, skonstruira nov

tajnopis $c'$, ki se dešifrira v čistopis, ki je na nek način povezan s čistopisom, ki pripada $c$. V splošnem velja, da je to neželena lastnost varnega kriptosistema, vendar bomo v poglavju 4.2 spoznali, da s tem, ko dovolimo gnetljivost kriptosistema, odpremo vrata popolnoma novim načinom uporabe kriptografije v oblaku.

# Poglavje 4

# Uporaba kriptografije v oblaku

Največji problem uporabe šifriranja v oblaku je v bistvu enak kot povsod, kjer poskušamo zagotoviti varnost. Potrebno je najti kompromis med zadovoljivim nivojem varnosti in uporabnostjo. Trivialno je narediti 100% varen sistem, vendar ta verjetno ne bo kaj prida uporaben.

Prav tako ločimo tri stanja digitalnih podatkov: podatke v mirovanju, podatke v gibanju in podatke v uporabi. V vsakem stanju obstajajo različne varnostne zahteve, zato so potrebni različni pristopi. Jaber in Zolkipli [16] sta uporabila tabelo 4.1, ki lepo ponazori načine uporabe kriptografije glede na različna stanja podatkov.

| | Podatki v mirovanju | Podatki v uporabi | Podatki v gibanju |
|---|---|---|---|
| Zaupnost | simetrične šifre | homomorfne šifre | TLS |
| Istovetnost | MAC | homomorfne šifre | TLS |
| Razpoložljivost | redundanca | redundanca | redundanca |

Tabela 4.1: Načini uporabe kriptografije v oblaku.

Zgodovinsko gledano je bila kriptografija uporabljena z motivom varovanja komunikacij, torej podatkov v gibanju, vendar na problem varovanja podatkov v mirovanju lahko gledamo kot na komunikacijo s samim sabo. To nam pred-

vsem poenostavi problem distribucije ključev, zato so za varovanje podatkov v mirovanju zelo primerne simetrične šifre.

Na začetku obdobja računalništva se je največ pozornosti posvečalo predvsem varnosti podatkov v mirovanju, saj so se informacijski sistemi nahajali znotraj podjetja, oziroma so bili do podjetja povezani preko najetih vodov. Prav tako so se podatki obdelovali na strojni opremi, ki je bila v lasti podjetja in se je nahajala na varni lokaciji. S prihodom storitev v oblaku, se je ta paradigma spremenila. Podatke je treba najprej prenesti v oblačni sistem, največkrat kar preko interneta. Prav tako podatke v oblak pošljemo z namenom, da jih bomo tam obdelovali in analizirali, kar pomeni da je potrebno poskrbeti tudi za varnost podatkov v uporabi.

To se odraža tudi v pristopih k varovanju podatkov v različnih stanjih. Dočim za varovanje podatkov v mirovanju obstajajo ustaljene prakse, pa za ostali dve področji velja, da se nenehno spreminjata. Na področju varovanja podatkov v gibanju, ki ga v primeru oblačnih sistemov večinoma pokrivata kriptografska protokola Secure Sockets Layer (SSL) in njegov naslednik Transport Layer Security (TLS), je v zadnjih letih prišlo do odkritij različnih napadov in posledično opustitve določenih verzij protokolov, ker te niso več zagotavljale zadovoljivega nivoja varnosti. Drugačna pa je situacija na področju varovanja podatkov v uporabi, saj v preteklosti ni bilo velike potrebe po praktični implementaciji takih kriptografskih protokolov, vendar se je s prihodom oblačnih storitev tudi to področje začelo pospešeno razvijati.

## 4.1 Funkcionalni kriptosistemi

Pred iznajdbo asimetričnih šifrirnih shem, je veljalo prepričanje, da če želimo vzpostaviti šifrirano komunikacijo, si morata najprej obe strani izmenjati skrivni ključ po varnem kanalu. V sedemdesetih letih sta Diffe in Hellman [9] to prepričanje postavila na glavo in predstavila algoritem za izmenjavo ključa, kjer ni potrebe po varnem kanalu. Podobno je še do pred kratkim veljalo

prepričanje:

1. šifrirano sporočilo je mogoče poslati samo eni osebi (prejemniku), ki si lasti pripadajoč dešifrirni ključ,

2. dešifriranje je možno samo na principu vse ali nič. Ali je prejemnik sposoben dešifrirati sporočilo v celoti (poseduje dešifrirni ključ), ali pa ga sploh ne more dešifrirati.

To lahko predstavlja nekaj nevšečnosti v oblačnih sistemih, kjer želimo uporabiti šifriranje. V primeru, da želimo deliti šifrirane datoteke, je potrebno za vsakega uporabnika ustvariti unikaten ključ in na novo šifrirati datoteko. V praksi se je sicer možno izogniti vsakokratnemu šifriranju, vendar bi bila veliko lepša rešitev, če bi lahko datoteko šifrirali tako, da jo lahko dešifrira več uporabnikov. Podoben primer so strežniški dnevniki in revizijske sledi, ki lahko vsebujejo občutljive podatke o uporabnikovi aktivnosti ob določenem času. Da uporabnikom zagotovimo visoko stopnjo zasebnosti, lahko take datoteke šifriramo. Vendar se pojavi problem, če policija po nalogu sodišča od ponudnika zahteva strežniške dnevnike skupaj z dešifrirnim ključem. Če je dešifriranje možno samo na principu vse ali nič, policija pridobi vpogled v aktivnost vseh uporabnikov sistema, ne le osumljenca. Rešitev bi bila, da za vsakega uporabnika izberemo unikaten ključ, vendar če želimo policiji omejiti tudi obdobje, za katerega lahko dešifrira podatke, je potrebno ta ključ rotirati in število teh ključev lahko naraste čez vse razumne meje. Priročnejša rešitev bi bila, če bi lahko uporabili tako šifrirno shemo, kjer lahko ključ dešifrira le del tajnopisa.

Izkaže se, da take šifrirne sheme res obstajajo, vendar je to področje kriptografije še dokaj novo in obstaja še veliko odprtih problemov. Čeprav so bili mnogi algoritmi, ki jih sedaj uvrščamo v to kategorijo, odkriti že prej, pa je bila prva formalizacija javno predstavljena šele leta 2011. V nadaljevanju bomo na kratko povzeli definicijo, kot je bila predstavljena v članku D. Boneh in drugih [7].

**Funkcionalnost**

**Definicija 4.1.** Funkcionalnost $F$, definirana na $(K, X)$ je funkcija

$$F : K \times X \to \{0,1\}^*,$$

predstavljena kot deterministično Turingov stroj. Množica $K$ je prostor vseh ključev in množica $X$ je prostor vseh čistopisov. Zahtevamo tudi, da množica $K$ vsebuje prazen ključ $\epsilon$.

**Funkcionalni kriptosistem**

**Definicija 4.2.** Funkcionalni kriptosistem za funkcionalnost $F : K \times X \to \{0,1\}^*$ je četverica algoritmov:

$$\begin{aligned} (\texttt{pp}, \texttt{mk}) &\leftarrow \texttt{setup}(1^\lambda), \\ \texttt{sk} &\leftarrow \texttt{keygen}(\texttt{mk}, \texttt{k}), \\ c &\leftarrow \texttt{enc}(\texttt{pp}, x), \\ y &\leftarrow \texttt{dec}(\texttt{sk}, c), \end{aligned}$$

kjer zahtevamo, da velja: $y = F(k, x)$.

`Setup` predstavlja postopek za generiranje javnih parametrov (`pp` - angl. public parameters) in glavnega zasebnega ključa (`mk` - angl. master key), $\lambda$ pa je varnostni parameter. Postopek `keygen` iz glavnega zasebnega ključa `mk` in ključa `k` ustvari skrivni ključ `sk`. Postopek `enc` je šifrirni postopek, ki sporočilo $x$ šifrira z javnim ključem `pp`. Dešifriranje je postopek `dec`, ki kot vhod vzame skrivni ključ `sk` in tajnopis $c$, kot rezultat pa vrne $y$, ki je pravzaprav rezultat funkcionalnosti $F$.

V funkcionalni šifrirni shemi za funkcionalnost $F(\cdot, \cdot)$, lahko upravnik sistema, ki si lasti glavni skrivni ključ, ustvari ključ `sk`, s katerim uporabnik izračuna vrednost funkcije $F(k, x)$ iz tajnopisa za $x$. Recimo, da je možno, da

za $F$ vzamemo katerokoli funkcijo, ki jo je možno opisati z deterministčnim Turingovim strojem. To nam odpre možnost, da v šifrirno shemo vgradimo "program" za nadzor dostopa in s tem zaobidemo omejitve, omenjene v točkah (1) in (2) na začetku poglavja.

Izkaže se, da je možno tudi povsem klasične asimetrične šifrirne sheme predstaviti v obliki funkcionalne šifrirne sheme, vendar so veliko bolj zanimive sheme, ki za ključe uporabljajo kar uporabnikovo identiteto - Identity Based Encryption (IBE) in njihova nadgradnja, ki je predstavljena v poglavju 4.1.

## Šifriranje na osnovi atributov in predikatov

Šifriranje na osnovi atributov in predikatov je poseben primer funkcionalne šifrirne sheme z vgrajeno kontrolo dostopa. Za začetek si poglejmo naslednji primer:

Anita ima datoteko $X$, ki bi jo rada delila s skupinama **Prijatelji** in **Znanci**. Prav tako ima datoteko $Y$, ki bi jo rada delila s skupinama **Družina** in **Prijatelji**. To lahko stori tako, da ustvari en ključ za datoteko $X$ in ta ključ deli s **Prijatelji** in **Znanci**, nato ustvari drug ključ za datoteko $Y$ in ta ključ deli s skupinama **Družina** in **Prijatelji**. Problem je, da mora Anita za vsako datoteko, ki jo želi deliti, ustvariti nov ključ.

Naivna rešitev tega problema bi bila, da Anita ustvari en ključ za vsako skupino ljudi, s katerimi želi deliti datoteke. Vendar življenje ni preprosto, in lahko se zgodi, da neka oseba pripada večim skupinam s katerimi želi Anita deliti datoteke. Prav tako si lahko zaželi deliti neko datoteko samo z ljudmi, ki so hkrati **Prijatelji** in **Sodelavci**. Recimo, da je to datoteka $Z$. Tako datoteko lahko najprej šifrira s ključem, ki pripada skupini **Prijatelji**, in nato še s ključem, ki pripada skupini **Sodelavci**. Naivno bi mislili, da je to dobra rešitev, vendar se izkaže, da tak sistem ni odporen proti zaroti. Šifrirano datoteko lahko kot posamezniki odprejo samo ljudje, ki pripadajo obema skupinama, saj imajo le oni v posesti oba ključa. Vendar pa se lahko proti Aniti zarotita dva posameznika, kjer noben od njiju ne pripada obema skupinama hkrati,

vendar imata skupaj dostop do obeh ključev, saj vsak od njiju pripada eni od skupin.

Možnost dešifriranja določajo pravila, ki so predstavljena z drevesi, kjer so notranja vozlišča logične operacije, listi pa predikati, ki so lahko zadovoljeni ali nezadovoljeni. Drevo beremo od listov proti korenu in za vsako notranje vozlišče ugotovimo ali je v zadovoljenem ali nezadovoljenem stanju. Na koncu pridemo do korena drevesa, in dešifriranje uspe samo v primeru, da je koren drevesa v zadovoljenem stanju.

Poznamo dve vrsti šifriranja na osnovi parametrov. V prvem je drevo, ki določa pravila shranjeno v ključu (angl. Key-policy attribute based encryption - KP-ABE), v drugem primeru pa je shranjeno v tajnopisu (angl. Ciphertext-policy attribute based encryption - CP-ABE).

**Funkcionalni kriptosistem s predikati**

**Definicija 4.3.** Velikokrat ima čistopis $x \in X$, določeno strukturo, recimo da je v obliki para $(\texttt{ind}, m) \in I \times M$, kjer imenujemo `ind` indeks in $m$ koristen tovor. Primer je lahko e-poštni sistem, kjer je indeks identiteta pošiljatelja, koristni tovor pa e-sporočilo.

Definirajmo predikat $P : K \times I \to \{0, 1\}$ in funkcionalnost $F : K \times X \to X \cup \perp$:

$$F(k, (\texttt{ind}, m)) = \begin{cases} m, & \text{če velja } P(k, \texttt{ind}) = 1, \\ \perp, & \text{če velja } P(k, \texttt{ind}) = 0 \end{cases}$$

Če tako funkcionalnost uporabimo v definiciji 4.2, to pomeni, da dešifrirni postopek $\texttt{dec}(\texttt{sk}, c)$, kjer je `sk` skrivni ključ za $k$ in $c = \texttt{enc}(\texttt{pp}, (\texttt{ind}, m))$ tajnopis, vrne čistopis, če in samo če $P(k, \texttt{ind}) = 1$.

Funkcionalni kriptosistem s predikati, podan v definiciji 4.3, lahko uporabimo za izgradnjo KP-ABE in CP-ABE kriptosistemov. Iz definicij 4.4 in 4.5

lahko vidimo da sta si sistema zelo podobna, razlika je le katere informacije hranimo v ključu $k$ in katere v funkcionalnosti $F$. V KP-ABE kriptosistemu nam ključ kodira formulo $\phi$ (pravilo) za dostop do podatkov, tajnopis pa vsebuje atribute, katerih vrednost je lahko `DA` ali `NE` (logična 1 ali logična 0). Dešifriranje v takem sistemu je uspešno le v primeru, da formula, podana v ključu, vrne rezultat 1 (`DA`) za dane atribute $\vec{z}$ v tajnopisu. Kriptosistem CP-ABE, definiran v 4.5, uporablja enak princip, le da so atributi kodirani v ključu, formula za dostop do podatkov pa v tajnopisu.

**KP-ABE kriptosistem**

**Definicija 4.4.** KP-ABE kriptosistem v $n$ spremenljivkah (atributih) je funkcionalni kriptosistem s predikatom iz definicije 4.3, kjer:

1. Prostor ključev $K$ je množica logičnih (Boolovih) formul $\phi$ v $n$ spremenljivkah $\vec{z} = (z_1, \ldots, z_n)$. Naj $\phi(\vec{z})$ označuje vrednost formule $\phi$ za $\vec{z}$.
2. Čistopis je par $(\texttt{ind} = \vec{z}, m)$, kjer je indeksni prostor $I : \{0,1\}^n$.
3. Predikat $P : K \times I \to \{0,1\}$ definiramo kot:

$$P(\phi \in K \setminus \{\epsilon\}, \texttt{ind} = \vec{z} \in I) = \phi(\vec{z}).$$

**CP-ABE kriptosistem**

**Definicija 4.5.** CP-ABE kriptosistem v $n$ spremenljivkah (atributih) je funkcionalni kriptosistem s predikatom iz definicije 4.3, kjer:

1. Prostor ključev $K := \{0,1\}^n$ je množica vseh vektorjev $\vec{z}$, ki predstavljajo točke, v katerih lahko izračunamo vrednost formule $\phi$.
2. Čistopis je par $(\texttt{ind} = \phi, m)$, kjer je indeksni prostor $I$ možica vseh logičnih formul v $n$ spremenljivkah.
3. Predikat $P : K \times I \to \{0,1\}$ definiramo kot:

$$P(\vec{z} \in K \setminus \{\epsilon\}, \texttt{ind} = \phi \in I) = \phi(\vec{z})$$

## 4.2 Homomorfni kriptosistemi

Homomorfizem je lastnost, da je določena algebraična operacija nad dvema čistopisoma, ekvivalentna neki drugi operaciji nad pripadajočima tajnopisoma. To pomeni, da namesto, da naredimo neko operacijo nad čistopisom, lahko izvedemo homomorfno operacijo nad tajnopisom in končni rezultat bo isti. Taki kriptosistemi bi v teoriji lahko omogočili varno računanje v oblaku, saj je možno vse operacije izvajati nad tajnopisi, tako da ni potrebe, da oblačni sistem kadarkoli pride v stik s čistopisom podatkov.

Potrebna lastnost homomorfne šifre je gnetljivost (angl. malleability), ki je sicer v večini ostalih shem nezaželena lastnost. Gnetljivost omogoča spremembo tajnopisa na tak način, da se ta dešifrira v čistopis, ki je na nek način povezan z originalnim čistopisom. Če imamo dan tajnopis za $m$, to pomeni, da je možno izračunati nov tajnopis, ki se bo dešifriral v vrednost $f(m)$ za neko funkcijo $f$, brez da bi sploh vedeli vrednost $m$.

Ločimo dva tipa homomorfnih šifer, glede na to, katere operacije lahko

opravimo nad tajnopisi. V prvo skupino spadajo delno homomorfni sistemi, kot so RSA brez dopolnjevanja, ElGamal in drugi. Taki sistemi so homomorfni samo za eno operacijo in kot taki ne prav zanimivi za splošno uporabo, je pa z njimi možno implementirati rešitev kot so varne e-volitve. Veliko bolj zanimivi pa so popolnoma homomorfni sistemi, kjer je nad tajnopisi možno izvesti poljubno operacijo. Prvi popolnoma homomorfni kriptosistem je v svoji doktorski disertaciji predstavil C. Gentry [11], vendar se njegova implementacija ni izkazala za najbolj praktično. Prvi problem predstavlja velikost javnega ključa, ki sega od 70 MB do 2,3 GB, odvisno od stopnje varnosti, ki jo želimo. Prav tako je problematična hitrost take sheme, saj obdelava enega bita informacije traja nekje med 30 sekundami in 30 minutami.

Za občutek lahko predstavimo nekoliko homomorfno šifrirno shemo iz [12]. Shema je veliko preprostejša od prve sheme v doktorski disertaciji C. Gentry, saj uporablja enostavno številsko aritmetiko, vendar je njena pomanjkljivost, da omogoča samo omejeno število ovrednotenj, preden rezultat postane neuporaben.

**Nekoliko homomorfen kriptosistem**

**Definicija 4.6.** Za izbran varnostni parameter $\lambda$, naj bo $N = \lambda$, $P = \lambda^2$ in $Q = \lambda^5$.

**Generiranje ključa -** `KeyGen`

Za ključ $p$ izberemo naključno liho $P$-bitno celo število.

**Šifriranje -** $e_k(m)$

Šifriranje $m \in \{0, 1\}$ poteka tako, da najprej izberemo naključno $N$-bitno število $m'$, za katerega velja $m = m' \mod 2$. Tajnopis je

$$c = m' + pq,$$

kjer je $q$ naključno $Q$-bitno celo število.

**Dešifriranje -** $d_k(m)$

Rezultat dešifriranja je

$$(c \mod p) \mod 2,$$

kjer velja:

$$c' = (c \mod p),$$

$$-p/2 < c' < p/2$$

in $p$ deli $c - c'$.

Tajnopisi v shemi, podani v definiciji 4.6, so blizu večkratniku števila $p$. Ostanek pri deljenju tajnopisa $c$ s številom $p$, $(c \mod p)$, imenujemo *šum*.

V taki shemi je možno tajnopise seštevati, odštevati in množiti po modulu 2. Ker seštevanje in množenje po modulu 2 hkrati predstavlja tudi logična vrata AND in XOR, lahko v takem sistemu ovrednotimo poljubno funkcijo, ki jo je možno predstaviti digitalno vezje. Problem nastane, ker vsaka od teh

operacij poveča količino "šuma", ki smo ga dodali pri šifriranju, in če ta naraste preko neke meje, sporočila ni več možno dešifrirati.

**Homomorfne operacije**

**Definicija 4.7.** V nekoliko homomorfnem kriptosistemu, podanem v definiciji 4.6, je možno izvesti naslednje operacije nad tajnopis:

**Operacije -** $\texttt{Add}(c_1, c_2)$**,** $\texttt{Sub}(c_1, c_2)$**,** $\texttt{Mult}(c_1, c_2)$

Seštevanje, odštevanje im množenje lahko izvedemo tako, kot smo navajeni. Torej $c_1 + c_2$, $c_1 - c_2$ in $c_1 \cdot c_2$ v $\mathbb{Z}_2$.

**Ovrednotenje -** $\texttt{Evaluate}(f, c_1, \ldots, c_t)$

Funkcijo $f$ predstavimo kot logično vezje, sestavljeno samo iz vrat XOR in AND. V aritmetiki modulo 2 logična vrata XOR predstavljajo seštevanje, logična vrata AND pa množenje, kar pa so ravno homomorfne operacije, definirane v prejšnji alineji.

Problem šuma je mogoče rešiti tako, da podatke dešifriramo. Če nam uspe dešifrirno funkcijo predstaviti kot logično vezje, ki ga je možno ovrednotiti v dani shemi, smo na dobri poti, da zmanjšamo šum. Kriptosistemom, ki so sposobni te operacije rečemo, da so *samozagonski* (angl. bootstrappable), in možno je pokazati [11], da iz takega sistema lahko sestavimo popolnoma homomorfen kriptosistem. Kljub temu, da ta postopek omogoča, da pridemo do popolnoma homomorfnega kriptosistema, pa je daleč od idealnega. Ovrednotenje dešifrirne funkcije, predstavljene kot digitalno vezje, je računsko zahteven postopek, prav tako pa tajnopis enega samega bita informacije zasede precej več prostora kot pripadajoč čistopis. Zato se v zadnjem času veliko raziskav usmerja v kriptosisteme, kjer *bootstraping* ni potreben. Predvsem se veliko pozornosti usmerja v problem *učenja z napakami* - Learning with Errors (LEW). Problem je prvič predstavil O. Regev [27] in izkazalo se je, da ta problem lahko predstavlja koristno osnovo za mnoge kriptografske konstrukcije. Čeprav pro-

blem šuma obstaja tudi v kriptosistemih, osnovanih na problemu LEW, pa je le tega možno zmanjšati na veliko hitrejši način kot je *samozagon*.

## 4.3 Varovanje podatkov v mirovanju

Podatke v mirovanju predstavljajo vsi podatki na trajnem pomnilniku, izključujoč podatke, ki so trenutno v gibanju, in podatke, ki so trenutno v delovnem pomnilniku računalnika. Praviloma so to podatki, ki se ne spreminjajo pogosto, oziroma arhivski podatki, ki se sploh ne spreminjajo. Količinsko predstavljajo glavnino vseh podatkov, kar pomeni, da morajo biti kriptografski algoritmi, ki delujejo na teh podatki, hitri. Zato so za varovanje takih podatkov zelo primerni simetrični kriptosistemi.

Podatki v mirovanju so dolgotrajne narave, kar pomeni, da je dolgotrajna tudi skrb za varnost šifrirnih ključev, ki nam zagotavljajo dostop do teh podatkov. V preteklosti so bili ključi "shranjeni" kar v uporabnikovi glavi, vendar je ob količini podatkov, s katerimi operiramo danes, to žal postala misija nemogoče. Dodaten problem predstavlja dejstvo, da je do podatkov pogosto potreben povsem avtomatski dostop, brez posredovanja človeka.

V kontekstu oblačnih sistemov je pri zagotavljanju varnosti podatkov v mirovanju najbolj problematičen del prav upravljanje s ključi, saj je potrebno poskrbeti za ključe velikega števila uporabnikov. Prav tako je s stališča varnosti neodgovorno, da ustvarjanje in upravljanje šifrirnih ključev zaupamo tretji osebi (ponudniku storitve). Zato je v primeru, da je potrebno zagotoviti visoko stopnjo varnosti, kljub temu pa želimo podatke hraniti v oblaku, najbolj primerno šifriranje podatkov, še preden jih pošljemo v oblak, tako imenovano *client-side* šifriranje. Prednosti so očitne, saj šifrirni ključ nikoli ne zapusti našega računalnika, tako da smo lahko prepričani, da do naših podatkov ne more dostopati nihče.

Kot smo že omenili, so za šifriranje takih podatkov najprimernejše simetrične šifre. Predvsem se priporoča uporaba modernih kriptosistemov, ki po-

| | hitrost (GB/s) |
|---|---|
| AES-128-GCM | 1,09 |
| CHACHA20-POLY1305 | 0,34 |
| SALSA20-256-SHA1 | 0,23 |

Tabela 4.2: Primerjava hitrosti treh priporočenih algoritmov. Test je bil izveden na procesorju Intel Core i5-2430M, ki podpira strojno pospešen AES

leg zasebnosti zagotavljajo tudi istovetnost podatkov. Primeri takih sistemov so `Salsa20-256-SHA1`, `Chacha20-Poly1305` in `AES-GCM`. `Salsa20` [3] in `Chacha20` [2] sta si si zelo podobni tokovni šifri (razlikujeta se le v krožni funkciji). Na drugi strani pa je bločna šifra `AES` v načinu delovanja `GCM`. `AES` je industrijski standard za šifriranje podatkov, kar se pozna tudi v dobri strojni podpori za pohitritev šifriranja. Večina sodobnih procesorjev na arhitekturi `x86` podpira nabor ukazov `AES-NI`, ki znatno pohitrijo postopke šifriranja, kot je razvidno v tabeli 4.2.

Pri uporabi bločnih kriptosistemov v kombinaciji z enim od načinom delovanja (kar velja v 99% primerov) se je potrebno zavedati posledic rojstnodnevnega paradoksa [34]. Glede na velikost bloka obstaja zgornja meja za količino podatkov, ki jih je še varno šifrirati z istim ključem. V kolikor to mejo presežemo, obstaja možnost, da napadalcu z uporabo rojstnodnevnega napada uspe dešifrirati delčke tajnopisa. Problem je še posebej pereč pri bločnih šifrah, ki uporabljajo 64 bitne bloke, saj je v tem primeru zgornja meja 32 GB, kar je v današnjih časih vse prej kot nedosegljiva količina podatkov. V primeru uporabe šifre s 128 bitnim blokom, se ta meja dvigne na 256 EB. Za šifro, ki uporablja $n$-bitne bloke, pa velja da je z njo varno šifrirati do $n * 2^{n/2}$ bitov.

**Rojstnodnevni napad na način delovanja CBC**

**Definicija 4.8.** V načinu delovanja CBC šifriranje poteka tako:

$$c_i = e_k(m_i \oplus c_{i-1}),$$

kjer za $c_{-1}$ uporabimo IV.
Po tem ko isti ključ uporabimo za šifriranje več kot $2^{n/2}$ blokov, kjer je $n$ velikost bloka v bitih, obstaja velika verjetnost da pride do trka. Trk je dogodek, kjer napadalec v tajnopisu opazi dva enaka bloka ($c_i = c_j$). Ker šifrirna funkcija $e_k$ izvede samo deterministično permutacijo nad vhodnimi podatki, lahko sklepamo, da je bil vhodni podatek $e_k$ za oba bloka enak ($m_i \oplus c_{i-1} = m_j \oplus c_{j-1}$). Iz tega sledi:

$$m_i \oplus m_j = c_{i-1} \oplus c_{j-1}.$$

Napadalcu sta že znana oba tajnopisa $c_{i-1}$ in $c_{j-1}$, zato lahko ugane bodisi vrednost $m_j$, bodisi vrednost $m_i$ in preko zgornje formule izračuna manjkajočo vrednost ter tako dešifrira delček sporočila.

## Konvergentne šifrirne sheme

V primeru oblačnih storitev, ki ponujajo hrambo podatkov v oblaku, pa je zaželena tudi sposobnost izločanja dvojnikov (angl. deduplication), saj ta ponudniku storitve omogoča prihranke pri količini pomnilnika, porabljenega za hrambo datotek. Izločanje dvojnikov lahko deluje na nivoju datoteke ali pa na nivoju blokov. Najpreprostejša implementacija je na nivoju datoteke. Ob nalaganju datoteke v sistem se preveri zgoščena vrednost (angl. hash) te datoteke, in v kolikor sistem ugotovi, da je datoteka z enako zgoščeno vrednostjo že shranjena v sistemu, samo poveča vrednost števca, ki kaže koliko kopij te datoteke že obstaja. Z uporabo tega principa lahko v sistemu obstaja praktično neomejeno število kopij iste datoteke, brez potrebe po neomejenih pomnil-

niških kapacitetah. Naprednejši načini izločanja dvojnikov datoteko najprej razbijejo na bloke konstantne ali spremenljive dolžine, in nato uporabijo isti postopek na blokih datoteke. Tako je možno doseči še večje prihranke prostora na diskih.

Težava pa nastane, ko želimo izločanje dvojnikov uporabiti skupaj s šifriranjem. Praviloma želimo, da tajnopis izda karseda malo informacij o čistopisu, ki mu pripada. V praksi to pomeni, da za dva tajnopisa ne moremo ugotoviti, ali pripadata istemu čistopisu (isti datoteki), saj se v postopku šifriranja pogosto doda naključni element (recimo IV v poglavju 3.1). Tak pristop sicer omogoča doseganje *semantične varnosti*, definirane v definiciji 4.9, vendar onemogoči znatne prihranke pomnilniškega prostora ob poskusu izločanja dvojnikov. V kolikor želimo imeti oboje, je potrebno znižati varnostne zahteve. Če teoretičnemu napadalcu dovolimo, da ugotovi, da dva tajnopisa pripadata istemu čistopisu (torej tak kriptosistem ni *semantično varen*) je izločanje dvojnikov možno.

**Definicija 4.9.** Kriptosistem je **semantično varen**, če iz tajnopisa ni mogoče na praktičen način pridobiti nobene informacije o čistopisu.

V principu je treba poskrbeti, da šifriranje enakega čistopisa vsakič da enak tajnopis. To pomeni, da moramo vsakič uporabiti enak šifrirni ključ in tudi IV, v kolikor ga dani kriptosistem zahteva. Lahko bi za vse datoteke vseh uporabnikov uporabili isti ključ in isti IV, vendar taka implementacija ne bi zagotavljala prav visoke varnosti. V veliko primerih ima šifriranje različnih čistopisov z enakim ključem in IV katastrofalne posledice za varnost. Tu nam pridejo na pomoč konvergentne kriptografske sheme. Prvo tako shemo je podjetje Stac, Inc. patentiralo že leta 1995, v zadnjih letih pa so predvsem zaradi vse večje uporabe oblačnih shramb podatkov te sheme deležne veliko pozornosti. Osnovna ideja sheme je, da se šifrirni ključ izračuna iz čistopisa. Ponavadi se izračuna zgoščena vrednost čistopisa, nato pa se iz te vrednosti na deterministično ustvari šifrirni ključ. Ker je zgoščena vrednost dveh enakih

datotek tudi enaka, in postopek za izračun ključa deterministično, smo tako dosegli cilj, da se dve isti datoteki vedno zašifrirata v isti tajnopis.

Vendar pa imajo zaradi znižanih varnostnih zahtev taki sistemi tudi svoje ranljivosti. Ker je postopek šifriranja deterministično, tako lahko napadalec, ki si lasti enako datoteko, ugotovi ali ima to datoteko tudi kateri drug uporabnik. Na prvi pogled to sicer ni videti tako problematično, vendar je na tak način mogoče ugotoviti, ali je uporabnik v sistem naložil piratsko vsebino ali pa prepovedano literaturo. V praksi se velikokrat uporabi izločanje dvojnikov na nivoju blokov, kar pa omogoča se bolj zvito izvedbo napada. V kolikor obstajajo datoteke, katerih vsebina je pretežno statična (recimo bančni izpiski), bodo tudi tajnopisi teh blokov v veliki meri enaki. Če ima napadalec dostop do podobnega bančnega izpiska, obstaja verjetnost da se tajnopis napadalčevega in žrtvinega dokumenta razlikujeta samo v parih blokih. Napadalec lahko uporabi surovo silo in ustvari vse možne bloke, nato pa uporabi prvi napad, da ugotovi, kateri bloki pripadajo dokumentu žrtve napada. Takim napadom se je možno izogniti z uporabo ustreznega nadzora dostopa, kot je opisano v članku [17].

## 4.4 Varovanje podatkov v gibanju

Za podatke v gibanju veljajo vsi podatki, ki se preko interneta pretakajo od uporabnika proti oblaku in obratno. Poleg teh so podatki v gibanju tudi vsi podatki, ki se prenašajo med različnimi strežniki znotraj oblačnega sistema.

Omrežja, po katerih podatki potujejo, velikokrat uporabljajo dinamične usmerjevalne algoritme, zato ne moremo vnaprej vedeti, po kateri poti bodo podatki potovali. Podatki v gibanju na svoji poti potujejo skozi veliko omrežnih naprav, ki imajo vpogled v podatke, prav tako pa jim ne moremo zaupati. Dodatno nevarnost predstavlja še napad vrinjenega napadalca, kjer napadalec izkoristi dinamičnost omrežja in se vrine na pot podatkov, med uporabnika in sistem.

Podatki so v gibanju praviloma malo časa, zato ni potrebe po kompleksnem upravljanju ključev. Z algoritmom za varno izmenjavo ključev, kot je Diffe-Hellman, opisan v poglavju 3.2, je možno, da se za vsako sejo uporabi nov, naključno generiran ključ.

Najbolj razširjen protokol za zaščito podatkov v gibanju je Secure Sockets Layer (SSL) in njegov naslednik Transport Layer Security (TLS), pogosto imenovana s skupnim imenom SSL/TLS. Poleg tega se predvsem za statične povezave med dvemi omrežji uporabljajo še protokoli iz družine VPN. Znotraj omrežja ponudnika se lahko uporabijo tudi tehnologije kot je VLAN. Slednji ne zagotavlja varnosti s pomočjo šifriranja, ampak deluje na principu omejevanja dostopa do prometa na skupnem mediju. Tak način pristopa sicer ščiti pred napadalci, ki imajo ustrezno malo pooblastil, recimo drugi uporabniki sistema, ne more pa nas zaščititi pred notranjimi napadalci, kot so zaposleni pri ponudniku storitve.

Ker do storitev v oblaku praviloma dostopamo s spletnim brskalnikom preko interneta, je standardna praksa uporaba protokola HTTP over TLS (HTTPS). HTTPS je implementacija HTTP protokola na povezavi, šifrirani s protokolom SSL ali TLS. Predvsem se priporoča uporaba slednjega, saj od leta 2014 naprej obstajajo učinkoviti napadi na vse različice protokola SSL, zato ta ne velja več za varnega.

Protokol SSL/TLS uporablja simetrične kriptosisteme za zagotavljanje zasebnosti, kriptografijo javnih ključev za ugotavljanje istovetnosti strežnika s katerim komuniciramo in za dogovor o ključu ter kode za overjanje, s katerimi zagotavlja celovitosti podatkov. Podrobnosti implementacije je možno najti v literaturi, kot je knjiga *Cryptography and network security* avtorja W. Stallings [30]. Ker protokol ne predpisuje točno določenih algoritmov, ampak prepušča izbor le teh uporabniku, je pomembno, da namenimo nekaj pozornosti pravilni nastavitvi protokola.

TLS se uporablja za zaščito občutljivih podatkov v transportu praktično na vsakem kotičku interneta, od varovanja gesel za prijavo na družabna omrežja,

varovanja e-poštnih sporočil med transportom do varovanja elektronskih plačil. Prav zato v zadnjih letih skorajda ne mine mesec, ko ni predstavljen nov napad na TLS, kljub temu pa TLS ob pravilnih nastavitvah še vedno velja za varnega.

## Priporočila za varno uporabo protokola SSL/TLS

Vsaka povezava, zaščitena s protokolom TLS najprej poizkusi overiti identiteto strežnika/osebe na drugi strani, saj se tako prepričamo, da res komuniciramo s tistim, s komer želimo. Če ne preverimo identitete druge strani, napadalcu omogočimo MITM napad, saj se lahko postavi med nas in strežnik, in tako prestreza komunikacijo. Za overjanje identitete se uporablja kriptografija javnih ključev, najpogosteje kar protokol RSA. Pomembno je, da uporabimo varne zasebne ključe ter tudi poskrbimo za varno hrambo le teh. Za večino primerov uporabe se priporočajo 2048 bitni RSA ključi, ki zagotavljajo stopnjo varnosti, primerljivo s 112 bitnim simetričnim ključem. V kolikor obstaja zahteva po višji stopnji varnosti, se lahko uporabijo 3072 bitni ključi, ki zagotavljajo približno 128 bitov varnosti. Vse te primerjave so zgolj informativne narave, saj se z napredkom napadov na RSA, stopnja varnosti, ki jo nudi določena dolžina ključa, vztrajno niža.

Poleg dolžine ključa je potrebno poskrbeti, da se za podpis digitalnega potrdila uporablja kriptografsko varna zgoščevalna funkcija. Od začetka leta 2016 `SHA1` ne smatra več za varno zgoščevalno funkcijo, zato je priporočena uporaba *SHA256*.

Družina protokolov `SSL/TLS` je sestavljena iz 5 različic protokolov: SSL v2, SSL v3, TLS v1.0, TLS v1.1 in TLS v1.2. Priporoča se uporaba najnovejše različice TLS v1.2, v kolikor pa je pa je potrebno zagotoviti podporo tudi za stranke, ki uporabljajo starejšo programsko opremo, pa je možna tudi uporaba TLS v1.0 in TLS v1.1.

**Pregled različic protokola SSL/TLS**

**SSL v2**

Protokol je zastarel, njegova uporaba je zelo nevarna. Zaradi napada `DROWN` [1], uporaba tega protokola ogrozi tudi varnost strežnikov, ki ne uporabljajo tega protokola, uporabljajo pa enak zasebni ključ kot ranljiv strežnik.

**SSL v3**

Uporaba SSL v3 v kombinaciji s protokolom HTTP zaradi napada `POODLE` [24] prav tako ni več varna.

**TLS v1.0**

Uporaba tega protokola sicer ni priporočljiva, vendar pogosto zaradi združljivosti s starejšo programsko opremo nujna. Glavna ranljivost je napad `BEAST`. Ranljivosti, ki so omogočale napad, so bile večinoma odpravljene v modernih brskalnikih, vendar vedno obstaja delež uporabnikov, ki ne uporablja posodobljene programske opreme.

**TLS v1.1**

Protokol velja za varnega, vendar podpira zastarele šifrirne algoritme, zato je potrebno biti pazljiv pri nastavitvah le teh.

**TLS v1.2**

Protokol prav tako velja za varnega, poleg tega pa ponuja še možnost uporabe overjenih šifrirnih postopkov, ki ponujajo višjo stopnjo varnosti.

Protokol `SSL/TLS` uporabniku omogoča izbor nabora kriptosistemov, ki se uporabljajo za doseganje zaupnosti, istovetnosti in celovitosti komunikacije. Žal nekateri kriptosistemi, ki jih protokol podpira, niso več varni za uporabo.

Za dogovor o ključu je priporočena uporaba sistemov, ki zagotavljajo prihodnjo varnost (angl. forward secrecy). Uporaba prihodnje varnosti pomeni, da

se za šifriranje prometa uporabljajo kratkoživi ključi, ki niso povezani z zasebnim ključem strežnika. Tako tudi v primeru, da pride do razkritja zasebnega ključa strežnika, napadalec ne more dešifrirati prometa. Priporoča se uporaba algoritmov na osnovi dogovora o ključu Diffie-Hellman, ponavadi jih najdemo pod kraticami `DHE` in `ECDHE`. Slednji algoritem namesto celoštevilskih obsegov uporablja aritmetiko na eliptičnih krivuljah, kar mu omogoča, da zagotovi isto stopnjo varnosti, ob uporabi krajših ključev kot `DHE`.

Pri izboru kriptosistema za šifriranje prometa se priporoča uporaba `AES`, v načinu delovanja `GCM`. Način delovanja `GCM` je podoben načinu delovanja `CTR`, predstavljenem v poglavju 3.1, s to razliko, da hkrati zagotavlja tudi istovetnost šifriranih podatkov. Izogniti se velja vsem bločnim kriptosistemom, ki uporabljajo bloke, krajše od 128 bitov, da se izognemo rojstnodnevnim napadom (opisani v definiciji 4.8), kot je `SWEET32` [4].

Prav tako se je potrebno izogniti kriptosistemu `RC4`, ki ne velja več za varnega in vsem tako imenovanim "izvoznim" kriptosistemom (angl. export ciphers). Te kriptosistemi izvirajo še iz časov hladne vojne, ko so v Združenih Državah veljali strogi izvozni zakoni za kriptografske sisteme, saj je vlada na njih gledala kot na orožje. Vlada se je bala, da bi močna kriptografija lahko prišla v neprave roke, zato je bilo potrebno kriptosisteme, ki so jih želeli izvoziti izven Združenih Držav Amerike, oslabiti.

## 4.5 Varovanje podatkov v uporabi

Podatki v uporabi so zelo kratkotrajne narave, vendar jih je kljub temu zelo težko zaščititi. Problem izhaja iz same definicije podatkov v uporabi, saj so to podatki, nad katerimi računalniški sistem želi izvesti neko operacijo oziroma jih prikazati uporabniku. Podatki v uporabi se nahajajo v delovnem pomnilniku računalnika, in so zato izpostavljeni tako imenovanemu hladnemu napadu (angl. cold-boot attack), kjer napadalec dobesedno zamrzne stanje pomnilnika ob delujočem računalniku in nato poskuša iz njega izvleči vsebino, predvsem

šifrirne ključe, saj lahko z njimi dešifrira veliko večjo količino podatkov, ki so trenutno v mirovanju ali gibanju.

Kljub temu pa tveganje za hladni napad ni veliko, saj z napadalčeve strani zahteva fizičen dostop do sistema in kar nekaj tehnologije. Večji problem predstavlja morebitna okužba sistema s korenskim kompletom [5], ki napadalcu omogoča dostop do celotnega pomnilnika sistema.

Vsem tem problemom se lahko izognemo, če podatki v obliki čistopisa, sploh nikoli ne pridejo v oblačni sistem. To lahko zagotovimo tako, da podatke šifriramo preden jih pošljemo v oblak. Vendar šifrirani podatki so po definiciji neločljivi od naključnega šuma. Dokler uporabljamo oblak samo kot priročno shrambo velikih količin podatkov, to ne predstavlja problema. Podatke šifriramo preden jih pošljemo v oblak. Kadar želimo do njih dostopati, pretočimo tajnopise nazaj iz oblaka na svoj, zaupanja vreden, računalnik, kjer jih dešifriramo. Na tak način smo preložili problem hrambe podatkov v oblak, vendar ostane nam še problem obdelave podatkov. V idealnem svetu bi želeli, da bi obstajal način, da pošljemo šifrirane podatke v oblak, ta pa bi nad njimi izvedel poljubno operacijo, brez da bi jih dešifriral.

V poglavju 4.2 smo spoznali homomorfne kriptosisteme, ki nam omogočajo prav to. Žal pa je to še precej novo področje kriptografije in taki kriptosistemi še niso primerni za splošno uporabo. Največji problem predstavlja hitrost izvajanja operacij nad šifriranimi podatki, pa tudi velikost tajnopisov. Nič čudnega ni, če v homomorfnem kriptosistemu iz 1 MB čistopisa dobite 10 GB tajnopisa. Seštevanje nad tajnopisi je sicer dokaj hitra operacija (manj kot milisekunda), vendar že preprosto množenje dveh bitov podatkov lahko vzame več sekund. Kljub temu, pa je možno homomorfne sisteme uporabiti za preprosto aritmetiko nad tajnopisi, kot so na primer *e-volitve* in v določenih drugih dobro definiranih problemih. Omeniti velja ekipo raziskovalcev iz Microsofta, ki je v začetku leta 2016 predstavila nevronsko mrežo, ki zmore več deset tisoč optičnih prepoznav znakov (OCR) na uro, kljub temu, da operira s homomorfno šifriranimi podatki.

# Poglavje 5

# Zaključek

Računalništvo v oblaku vsekakor ni več stvar prihodnosti, ampak je že tu. Lahko trdimo, da večina uporabnikov storitve v oblaku uporablja, ne da bi poznali varnostna tveganja povezana z njihovo uporabo. Bolj zadržani so tehnično bolje podkovani uporabniki in podjetja, ki se zavedajo večine tveganj, ki jih prinese uporaba oblačnih storitev. Kot je bilo omenjeno, "100% varnost" sicer obstaja v teoriji, vendar jo je v praksi praktično nemogoče implementirati, zato kot uporabnik oblačnih storitev ne smemo nasedati foskulam, kot so *"Secured by AES"* ali pa *"Secured by RSA"*, saj nam te trditve brez točnih tehničnih podrobnosti implementacije ne zagotavljajo ničesar. Varnost je širok pojem, obstaja veliko načinov, kako jo zagotoviti, in še več načinov, kako jo zaobiti ali izničiti. Na tem mestu velja opomniti, da nikoli, ampak res nikoli ne poskušajte izumiti svojega šifrirnega postopka. Mnogi so poskusili in le redkim je uspelo. Varnost mora sloneti na rigoroznih matematičnih dokazih, ne na občutku, da nekaj izgleda varno. Predvsem je potrebno upoštevati Kerckhoffovo načelo, da mora biti sistem varen, če sovražnik pozna vse njegove podrobnosti, le ključa ne.

# Slike

# Tabele

# Literatura


[1] N. Aviram, S. Schinzel, J. Somorovsky, N. Heninger, M. Dankel, J. Steube, L. Valenta, D. Adrian, J.A. Halderman, V. Dukhovni, et al. DROWN: Breaking TLS using SSLv2. 51

[2] D.J. Bernstein. ChaCha, a variant of Salsa20. Objavljeno v *Workshop Record of SASC*, del 8, 2008. 45

[3] D.J. Bernstein. The Salsa20 family of stream ciphers. Objavljeno v *New stream cipher designs*, strani 84–97. Springer, 2008. 45

[4] K. Bhargavan in G. Leurent. On the Practical (In-)Security of 64-bit Block Ciphers. *Spletni vir*, 2016. Dostopno na `https://sweet32.info/SWEET32_CCS16.pdf`. 52

[5] Bill Blunden. *The Rootkit arsenal: Escape and evasion in the dark corners of the system*. Jones & Bartlett Publishers, 2012. 53

[6] D. Boneh, G. Di Crescenzo, R. Ostrovsky, in G. Persiano. Public key encryption with keyword search. Objavljeno v *International Conference on the Theory and Applications of Cryptographic Techniques*, strani 506–522. Springer, 2004.

[7] D. Boneh, A. Sahai, in B. Waters. Functional encryption: Definitions and challenges. Objavljeno v *Theory of Cryptography Conference*, strani 253–273. Springer, 2011. 35

[8] R. Chandramouli, M. Iorga, in S. Chokhani. Cryptographic key management issues and challenges in cloud services. Objavljeno v *Secure Cloud Computing*, strani 1–30. Springer, 2014.

[9] W. Diffie in M. Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976. 29, 34

[10] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, in J. Wernsing. CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy. Technical report, February 2016.

[11] C. Gentry. *A fully homomorphic encryption scheme*. Doktorska dizertacija, Stanford University, 2009. 41, 43

[12] C. Gentry. Computing arbitrary functions of encrypted data. *Communications of the ACM*, 53(3):97–105, 2010. 41

[13] S. Goldwasser, Y. T. Kalai, R.A. Popa, V. Vaikuntanathan, in N. Zeldovich. How to run turing machines on encrypted data. Objavljeno v *Advances in Cryptology–CRYPTO 2013*, strani 536–553. Springer, 2013.

[14] S. Goldwasser in S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, apr 1984. Dostopno na `http://dx.doi.org/10.1016/0022-0000(84)90070-9`.

[15] D. Harnik, B. Pinkas, in A. Shulman-Peleg. Side channels in cloud services: Deduplication in cloud storage. *IEEE Security & Privacy*, 8(6):40–47, 2010.

[16] A. N. Jaber in M. F. B. Zolkipli. Use of cryptography in cloud computing. Objavljeno v *Control System, Computing and Engineering (ICCSCE), 2013 IEEE International Conference on*, strani 179–184. IEEE, 2013. 33

[17] S. Keelveedhi, M. Bellare, in T. Ristenpart. DupLESS: server-aided encryption for deduplicated storage. Objavljeno v *Presented as part of the*

*22nd USENIX Security Symposium (USENIX Security 13)*, strani 179–194, 2013. 48

[18] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, in B. Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. Objavljeno v *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, strani 62–91. Springer, 2010.

[19] J. Li, X. Chen, M. Li, J. Li, P. PC Lee, in W. Lou. Secure deduplication with efficient and reliable convergent key management. *IEEE transactions on parallel and distributed systems*, 25(6):1615–1625, 2014.

[20] G. McGraw. *Software security: building security in*, del 1. Addison-Wesley Professional, 2006.

[21] J.D. Meier, A. Mackman, M. Dunner, S. Vasireddy, R. Escamilla, in A. Murukan. Improving web application security: threats and countermeasures. *Microsoft Corporation*, 3, 2003. 6

[22] P. Mell in T. Grance. The NIST definition of cloud computing. 2011. 2

[23] A. J. Menezes, P. C. Van Oorschot, in S. A. Vanstone. *Handbook of applied cryptography*. CRC press, 1996.

[24] B. Möller, T. Duong, in K. Kotowicz. This POODLE bites: exploiting the SSL 3.0 fallback. *Spletni vir*, 2014. 51

[25] S.N. Molotkov. Quantum cryptography and VA Kotel'nikov's one-time key and sampling theorems. *Physics-Uspekhi*, 49(7):750–761, 2006. 18

[26] C. Ramaswamy, I. Michaela, in C. Santosh. Cryptographic Key Management Issues and Challenges in Cloud Services. Objavljeno v *Secure Cloud Computing*, strani 1–30. Springer Science + Business Media, dec 2013. Dostopno na `http://dx.doi.org/10.1007/978-1-4614-9278-8_1`.

[27] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):34, 2009. 43

[28] M. Schwartz. New virtualization vulnerability allows escape to hypervisor attacks. *Information Week Security. Luettu*, 4:2013, 2012. 12

[29] C.E. Shannon. Communication theory of secrecy systems. *Bell system technical journal*, 28(4):656–715, 1949. 18

[30] W. Stallings. *Cryptography and network security: principles and practices.* Pearson Education India, 2006. 49

[31] D. R. Stinson. *Cryptography: Theory and Practice, Third Edition.* Discrete Mathematics and Its Applications. Taylor & Francis, 2005. Dostopno na `https://books.google.si/books?id=uhl_kYfpgo4C`.

[32] M. W. Storer, K. Greenan, D. DE Long, in E. L Miller. Secure data deduplication. Objavljeno v *Proceedings of the 4th ACM international workshop on Storage security and survivability*, strani 1–10. ACM, 2008.

[33] G.S. Vernam. Secret signaling system, Julij 22 1919. US Patent 1,310,719. Dostopno na `https://www.google.com/patents/US1310719`. 18

[34] Wikipedia. Birthday problem, 2016. [Spletni vir; dostopano 22-Avgust-2016]. Dostopno na `https://en.wikipedia.org/wiki/Birthday_problem`. 45

[35] Saman Taghavi Zargar, James Joshi, in David Tipper. A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks. *IEEE Communications Surveys & Tutorials*, 15(4):2046–2069, 2013. 6

[36] Y. Zhang, A. Juels, M.K. Reiter, in T. Ristenpart. Cross-VM side channels and their use to extract private keys. Objavljeno v *Proceedings of the 2012 ACM conference on Computer and communications security*, strani 305–316. ACM, 2012.